IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
MIDLAND/ODESSA DIVISION

INTELLECTUAL VENTURES I LLC and INTELLECTUAL VENTURES II LLC,	§	
Plaintiffs,	§	Civil Action No. 7:24-cv-00277-ADA
v.	§	JURY TRIAL DEMANDED
SOUTHWEST AIRLINES CO.,	§	
Defendant.	§	

DEFENDANT SOUTHWEST AIRLINES CO.'S
PRELIMINARY INVALIDITY CONTENTIONS

Pursuant to the Court's Standing Order Governing Proceedings ("OGP"), Defendant Southwest Airlines Co. ("Defendant" or "Southwest") submits the following Invalidity Contentions regarding United States Patent Nos. 8,332,844 (the '844 Patent), 8,407,722 (the '722 Patent), 7,949,785 (the '785 Patent), 8,027,326 (the '326 Patent), 7,324,469 (the '469 Patent), and 7,257,582 (the '582 Patent) (collectively, the "Asserted Patents"). Southwest makes these Invalidity Contentions based on its present understanding of Plaintiffs Intellectual Ventures I LLC's and Intellectual Ventures II LLC's ("Plaintiffs" or "IV") Preliminary Infringement Contentions (the "Infringement Contentions") and the contended scope of the asserted claims.

The currently asserted claims are identified in the table below (collectively, the "Asserted Claims").

Patent	Asserted Claims[1]
'844 Patent	7 and 11
'722 Patent	14, 16, 17, and 18
'785 Patent	30, 35, and 37
'326 Patent	1, 4, and 18
'469 Patent	24, 25, 26, 28, and 32
'582 Patent	1, 2, 5, and 7

I. PRELIMINARY STATEMENT

These invalidity contentions are preliminary in nature and do not address claim construction or noninfringement. Southwest reserves all rights with respect to such issues, including its position that claims of the Asserted Patents should be construed in a particular manner and/or are not infringed. These invalidity contentions are based on Southwest's current knowledge, understanding, and belief of the Asserted Patents and prior art, of Plaintiffs' infringement theories (as disclosed in the Infringement Contentions), and of the facts and other information available as of the date of these invalidity contentions. Southwest's investigation, discovery, and analysis of information related to this action are ongoing. Additional discovery and/or orders of the Court may require Southwest to amend or supplement these invalidity contentions, and Southwest reserves the right to do so as the case proceeds. These contentions represent Southwest's good-faith effort to provide a comprehensive identification of prior art relevant to this case, but Southwest reserves the right to modify or supplement its prior art list and invalidity contentions at a later time with, or based upon, pertinent information that may be subsequently discovered.

[1] Plaintiffs' Disclosure of Preliminary Infringement Contentions, served February 10, 2025.

A. No Waiver

Nothing in these invalidity contentions is intended, nor should be construed, as a waiver of any claim construction argument, noninfringement position, or argument under Pre-AIA 35 U.S.C. § 101, 102, 103, and/or 112. Southwest's statements herein (including the accompanying claim charts) reflect Southwest's present understanding of the purported scope of the claims as alleged by Plaintiffs in their Infringement Contentions. For the avoidance of doubt, these contentions are not intended to present or argue in favor of any construction of the Asserted Claims, and Southwest reserves the right to present constructions that differ from any construction that may be alleged to underlie these contentions. These invalidity contentions do not constitute an acquiescence to Plaintiffs' interpretation of any claims to the extent they depart from the plain meaning of the Asserted Claims and from the proper scope of the claims as may be subsequently construed by the Court. Southwest disagrees with Plaintiffs' positions as to the purported scope of the Asserted Claims to the extent they seek to expand the scope of the Asserted Claims beyond their proper construction.

The Asserted Claims have not yet been construed by the Court in this case and, thus, Southwest has not yet had the opportunity to compare the Asserted Claims of the Asserted Patents (as construed by the Court) with the prior art. Southwest reserves the right to amend, supplement, or materially modify its invalidity contentions after the claims have been construed by the Court in this case. Southwest also reserves the right to amend, supplement, or materially modify its invalidity contentions in response to any claim construction positions that Plaintiffs may take, and to assert that a claim is indefinite, not enabled, or fails to meet the written description requirement of Pre-AIA 35 U.S.C. § 112 based on any claim construction position Plaintiffs may take in this case or based on any claim construction the Court may adopt in this case.

B. No Admission

Nothing disclosed herein is an admission or acknowledgement that any product accused of infringement by Plaintiffs in its Infringement Contentions (the "Accused Products"), or any of Southwest's other products or services, infringes any of the Asserted Claims.

Plaintiffs do not currently assert that any claim limitation is met under the doctrine of equivalents. However, to the extent that Plaintiffs are later permitted by the Court to amend its contentions to pursue any currently undisclosed doctrine of equivalents theories, Southwest reserves the right to supplement or amend these invalidity contentions to account for such amendments.

To the extent that any of the prior art references disclose the same functionality or feature of any of the Accused Products, Southwest reserves the right to argue that said feature or functionality does not practice any element of any of the Asserted Claims, and to argue, in the alternative, that if said feature or functionality is found to practice any element of any of the Asserted Claims, then the prior art reference demonstrates that the element is not novel, is obvious, and/or is otherwise not patentable.

Attached hereto are representative claim charts that demonstrate how the Asserted Claims of the Asserted Patents are invalid in view of certain prior art. The references cited in the attached claim charts may disclose the limitations of the Asserted Claims expressly and/or inherently. The suggested obviousness combinations may be presented in conjunction with or in the alternative to any contentions regarding anticipation. These obviousness combinations should not be construed to suggest that any reference included in any combination is not anticipatory in its own right.

C. Reservation of Rights

Prior art not included in this disclosure may become relevant. Southwest is currently unaware of the extent, if any, to which Plaintiffs will contend that limitations of the Asserted

Claims are not disclosed in the prior art identified by Southwest. Southwest reserves the right to identify other references that would have made the addition of the allegedly missing limitation to the disclosed device or method obvious or show that the allegedly missing limitation would have been known or readily apparent to one of ordinary skill in the art at the time of the invention in light of the disclosure of the prior art at issue.

Southwest provides these invalidity contentions only for the claims asserted by Plaintiffs, but reserves the right to seek invalidation of all claims in the Asserted Patents.

D. Ongoing Investigation

Southwest's investigation is ongoing, and Southwest reserves the right to amend its disclosures and document production to account for evidence uncovered as its investigation continues. Such amendments include identifying and relying on additional references that may result from Southwest's further search and analysis. Southwest reserves the right to supplement these contentions in light of any additional prior art of which Plaintiffs are aware and did not disclose to Southwest in discovery, or that might be subsequently disclosed by Plaintiffs in response to Southwest's discovery requests. Southwest anticipates issuing subpoenas to third parties believed to have knowledge, documentation and/or corroborating evidence concerning some of the prior art listed herein and/or additional prior art. These third parties include, but are not limited to, the authors, employers of authors, inventors, assignees, or former or current employees of assignees, of the references and/or products identified in these invalidity contentions. Southwest reserves the right to supplement these contentions in light of any newly discovered information produced by these or other third parties from which Southwest may seek discovery.

Additionally, because discovery is ongoing, Southwest reserves the right to present additional prior art references and/or disclosures under 35 U.S.C. §§ 101, 102, and/or 103, located during the course of such discovery or further investigation, and to assert invalidity under Pre-AIA

35 U.S.C. §§ 102(c), (d), or (f), to the extent that such discovery or investigation yields information forming the basis for such invalidity.

Based on Southwest's investigation and knowledge developed to date, and pursuant to the Court's OGP, Southwest: (1) serves, attached as exhibits hereto, charts setting forth where in the prior art references each element of the Asserted Claims are found, (2) identifies any limitations Southwest contends are indefinite or lack written description under § 112, and (3) identifies any claims Southwest contends are directed to ineligible subject matter under § 101. Southwest's § 101 contentions (1) identify the alleged abstract idea, law of nature, and/or natural phenomenon in each challenged claim; (2) identify each claim element alleged to be well-understood, routine, and/or conventional; and (3) to the extent not duplicative of §§ 102/103 prior art contentions, prior art for the contention that claim elements are well-understood, routine, and/or conventional. Southwest has produced with these invalidity contentions the prior art referenced in the invalidity contentions. As for technical documents, including software where applicable, sufficient to show the operation of the accused product(s), IV has accused Southwest of using certain open-source software programs that are not owned or controlled by Southwest but are publicly available here.

https://docs.docker.com/build-cloud/

https://kubernetes.io/docs/home/

https://kafka.apache.org/documentation/

https://hadoop.apache.org/docs/current/

https://spark.apache.org/docs/latest/

IV has not accused specific implementations of those programs at Southwest. To the extent that IV does at some point identify specific uses or implementations at Southwest, Southwest will supplement the production to include any additional available technical details, if any. For the '469

and ‘326 patents, Southwest has produced its contracts but does not have technical documents. Those documents are in the possession, custody and control of Viasat and Anuvu.

II. THE ASSERTED PATENTS

Southwest sets forth a summary of its current understanding of the state of the art for the general subject matter of the Asserted Patents. Southwest reserves the right to rely on each of the prior art references disclosed below and charted in the attached Exhibits. Southwest also reserves the right to rely on the discussions of the state of the art and prior art for the Asserted Patents and their respective file histories, including any prior art cited in the file histories, in explaining the state of the art and the ways that each respective reference corresponds with the Asserted Claims. Southwest further expressly reserves the right to supplement its summary of the state of the art, including for example, by information from any of the authors, named inventors, or assignees of any of the prior art references, by personnel familiar with products based on any of the prior art, or by technical experts retained on behalf of any party.

A. Overview of the ‘844 Patent

The ‘844 Patent initially describes clustered computing involving multiple compute nodes with Operating System (“OS”) software image management, where nodes are activated by diskless booting protocols or remote software installation to local storage, and where a boot image is required for each node in the cluster – and states that storing a boot image for each server results in wasted storage, and copying the entire master image at boot results in long boot times. ‘844 Patent, 1:31-67. The Background also discusses branching-store file systems where a read-only “root” image is accessible by all nodes, and changes made by a node to the root image are stored in a “leaf” image unique to that node – with a filter operating between the nodes and the OS file system(s) that merges the leaf images with the root image and delivers the result to the appropriate

node (which the node views as its own cohesive instance of the application environment). *Id.* 2:13-24.

The 844 patent's system, as shown in Fig. 1, includes a general purpose computing system environment, such as compute node 100, which includes at least one processing unit 102 and memory 104). *Id.* 4:27-36. Nodes may also have additional features, including additional removable 108 and/or non-removable 110 storage, communications connections 112, input devices 114, and output devices 116. *Id.* 4:36-5:10. Memory 104, removable storage 108, and non-removable storage 110 are all computer storage media, which includes "any…medium which can be used to store the desired information and which can be accessed by compute node 100." *Id.* 4:46-55.

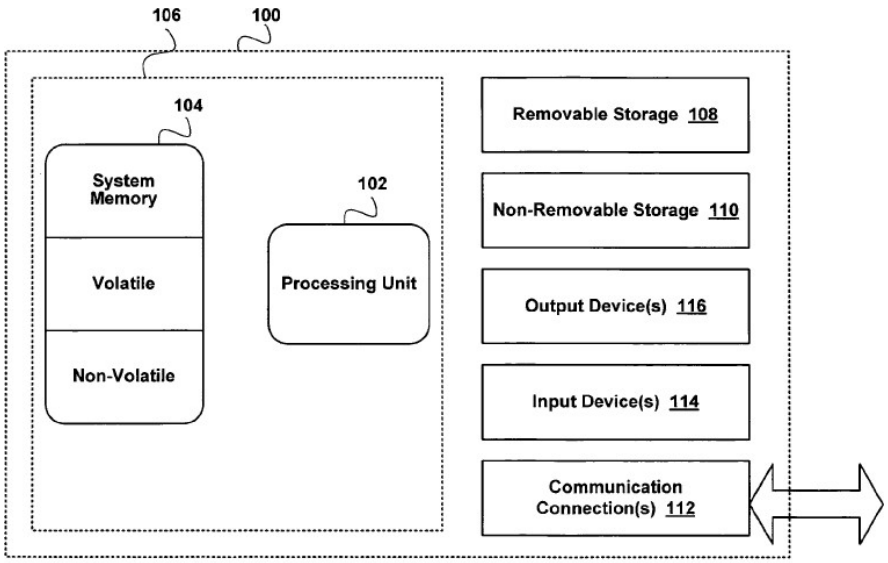


FIG. 1

Fig. 2 diagrams system 200 (e.g., a multi-computer system) having a number of compute nodes 220a-n coupled to a first storage unit 240 and a corresponding second storage unit 250a-n through a corresponding union block device ("UBD") 230a-n. '844 Patent, 5:12-22. To the

compute nodes 220a-n, it appears that they have access to their own version of the application environment. However, a separate boot image is not created and stored for each node. *Id.* 5:22-2.

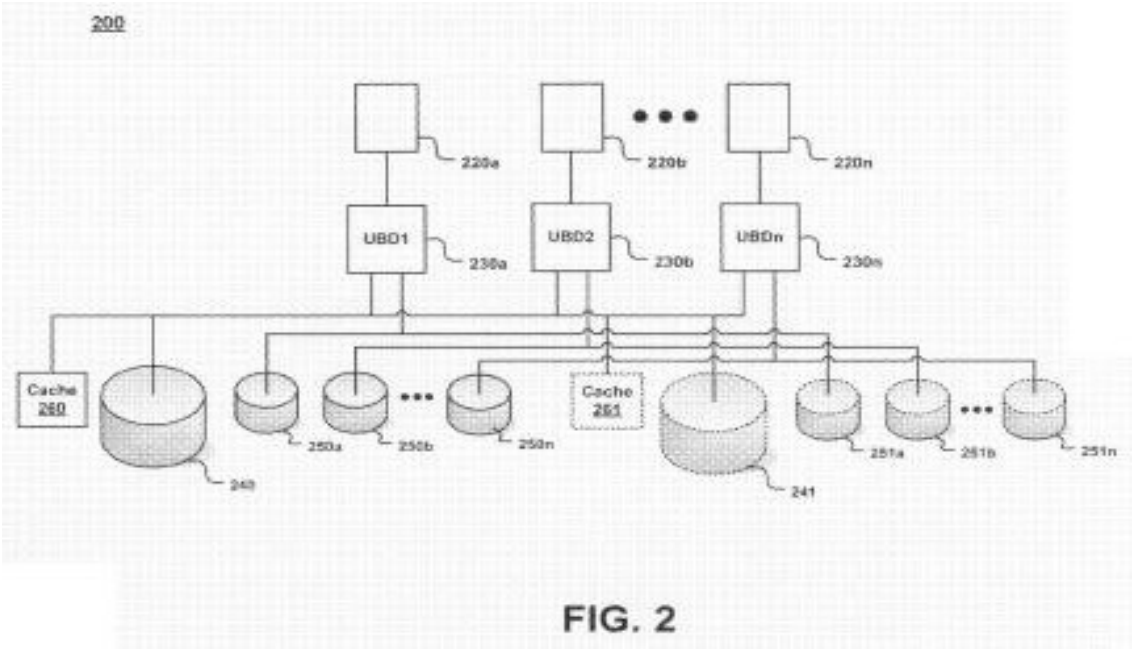


FIG. 2

Instead, a first storage unit 240 stores blocks of the root image. The root image contains data initially common to the compute nodes 220a-n, and is not changed by them. *Id.* 5:27-32. Each compute node 220a-n also has access to a corresponding second storage unit 250a-n for storing a leaf image, which in the challenged claims includes "only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of [said/the] root image…" or similar language. *Id.* 5:33-34, 10:61-65, 11:32-35, 12:4-8, 12:47-50. The first storage unit 240 and second storage units 250a-n may each be contained on separate physical storage devices, on separate logical spaces on the same storage device, or any combination thereof. *Id.* 5:34-38.

A compute node 220a-n mounts its instantiation of the application environment via its respective UBD 230a-n which, in one embodiment, are "effectively low level drivers that operate as an interface between the first and second storage devices and the file system of each compute

node 220a-n." *Id.* 5:59-64. UBDs 230a-n determine what "leaf image" (from the appropriate second storage unit 250) is needed for portions of the image that their respective compute nodes 220a-n have changed, and locate the portions of the image not changed by their respective node (which may reside in the root image or intermediate images comprising versions of the root image, not depicted in Fig. 2). '844 Patent, 6:4-13. UBDs 230a-n also create a new leaf image on a respective second storage unit 250a-n when their respective compute nodes 220a-n make changes to their instantiations of the image (and may also modify the leaf image in response to their respective compute node's access to its instance of the image). *Id.* 6:13-22.

The blocks of the root image, leaf image, and cache (discussed below) are merged "on the fly" to create the node's image. *Id.* 8:13-15. From the node's perspective, the created image appears as one cohesive image even though a separate and complete image is not actually stored for the node. *Id.* 8:18-55. This saves space without significantly impacting load times. *See id.* 2:51-57.

The '844 patent also claims to further improve speed by caching portions of the root image in a cache 260 (as several nodes in a cluster may often access the same root image data). '844 Patent, 2:59-61, 6:38-43. This may shorten times to load blocks by as much as 90%. *Id.* 6:43-46. Because cache capacity is typically limited, older data can be removed (as the amount approaches a threshold) to make way for new data by techniques "known in the art." *Id.* 7:1-16.

The 844 patent also provides an "added benefit" by which one compute node can index the contents of the (shared) root image and provide the results to the other compute nodes (e.g., by storing results on a shared storage unit or providing the results directly). *Id.* 7:23-31.

B. Overview of the '722 Patent

The '722 patent "pertains ... to transferring information for remotely updating content at client devices through the digital networks." '722 Patent, 1:24-27. For example, claim 14 recites:

14. A method comprising:

providing, using a processing device of an input source, a data representation to a client device, different from the input source, coupled to a routing network, wherein the data representation includes at least one live object recognizable by the client device, and causing the client device to respond to the live object of the data representation by determining an object identifier (ID) of the live object and to register for updates of the live object with the routing network, such that registering the client device with the routing network provides client connection information to a node in the routing network; and

sending, using the processing device of the input source, an update message to the routing network, wherein the update message identifies the live object and contains update data that updates a property of the live object,

wherein a gateway device at the routing network is configured to identify a category of the update message based on the input source, to determine a node type to which the identified category maps, and to route the update message to the node, having the node type, at the routing network,

wherein the node is configured to identify the client device as a registered device and to route the update message to the client device, and

wherein the client device processes the update message upon receipt to update the property of the live object at the client device.

Fig. 1 (annotated below) illustrates the input source (e.g., dynamic content provider 116, information provider 108, server 112 alone or combined), client device 114 (e.g., PC, phone, PDA, web-browsing appliance, other devices), routing network 110, and registry 125 identifying clients who have registered for live objects, and a "web page" (which can be any text, data, and/or programs) to clients 114 over communication links 117 (e.g., network, Internet). *Id.* at 4:11-12, 4:55-5:9, 6:19-30, 6:53-60, 10:14-23. A few servers and clients are shown in Fig. 1, but there could be thousands or even millions. '722 Patent, 4:60-63.

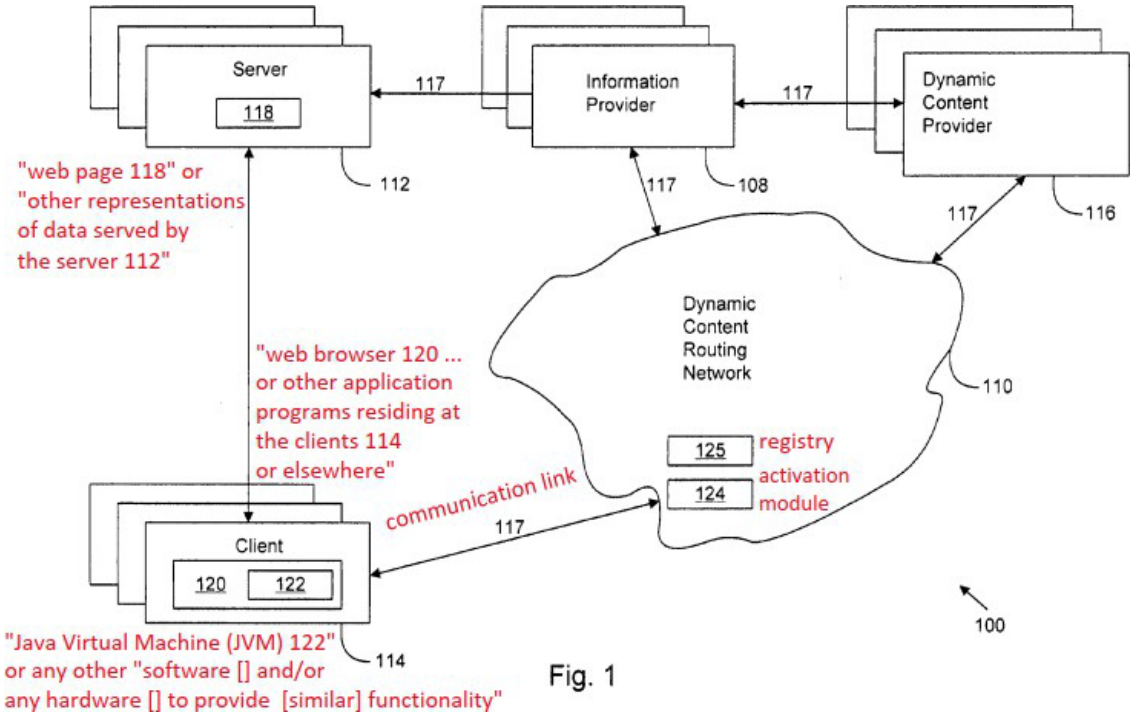


Fig. 1

The "routing network [] routes messages [from the input source (e.g., information/content provider or its server)] containing data for updating properties of live objects to clients displaying web pages or other representations of data." *Id.* at 3:9- 12, 6:22-29; *see also* 11:21-43, 12:24-25, Figs. 2, 3. "The web server that initially provides the web pages to the clients does not need to track which clients are currently displaying the live objects. Instead, the ... input source [] ... simply sends an update message to the routing network" and "the update messages are provided to the clients only when the live object changes." *Id.* at 3:12-18. The "'live objects,' which are designated to be real-time dynamically-updateable objects[, are] identified by an object identifier, or object ID." 5:28-31. The "objectID [] is preferably a string. The string can encode any information ... [and can be] a simple textual and/or numeric identifier" (e.g., "Bang$" for the stock symbol "BANG"). *Id.* at 8:34-37, 8:62-65.

"The clients 114 contact the routing network 110 and register for update messages for the objectIDs [of the live objects] on the web page[/'other application programs residing at the clients 114 or elsewhere']. The routing network 110, in turn, preferably maintains a registry indicating which clients have registered 35 for which object IDs." '722 Patent, 5:32-36, 6:58-59. Input sources send update messages "any time the[y] want[] to update a property of a live object. Each update message preferably identifies a live object [by object ID] and contains data for updating a property of the identified live object. The routing network 110 accesses the registry and determines which clients have registered for the identified object. Then, the routing network 110 routes the update message to the appropriate clients. Upon receipt of an update message, the clients 114 update the specified property of the live object." *Id.* at 5:41-48, 9:59-62. The "property" to update "include[s] any modifiable data related to the [live] object or referenced with respect to the object" (e.g., updates to "stock prices and/or sports scores" or any other data at the client which "may be a simple

numerical or textual value, e.g., '4'"). *Id.* at 7:23-24, 8:57-63, 9:53-56. Fig. 4 shows two differently-formatted webpages 410, 412 with the same "scores object 416" set to the score of the game "42" to "7." *Id.* at 12:37-46, 13:3-18.

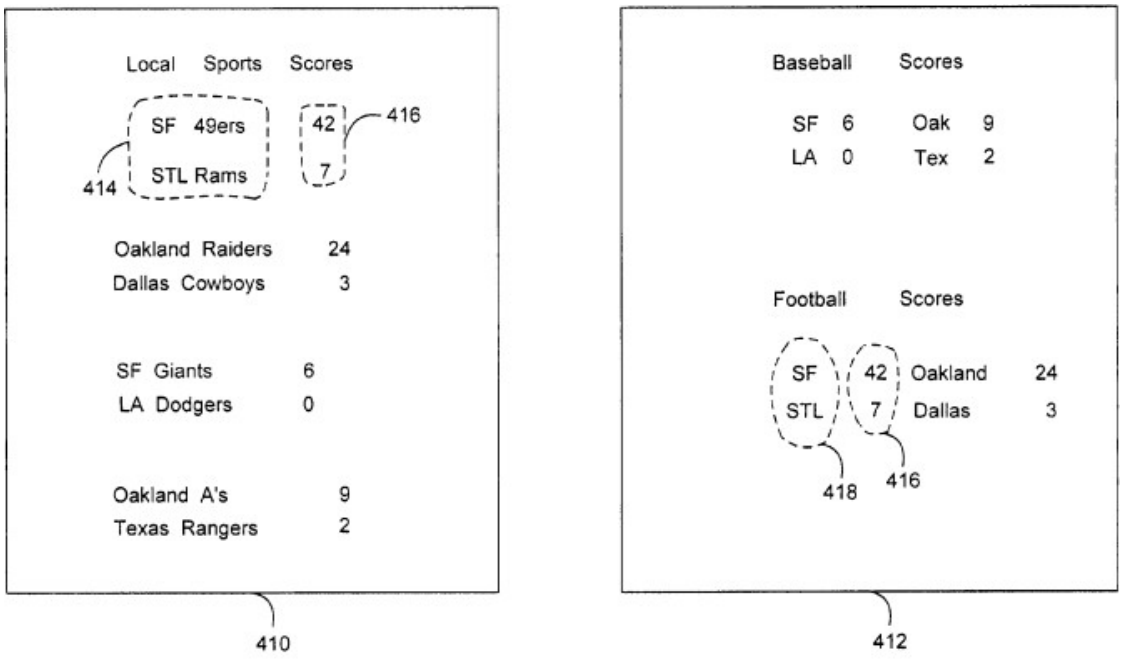


Fig. 4

"[T]o update the score of ... the game on both web pages, the input source 210 simply sends an update message to the routing network 110 specifying the objectID for the scores object 416 and the update to the score property. The routing network 110 routes the update message to the appropriate clients 114, and the clients update the appropriate score regardless of the particular page layout." *Id.* at 13:18-25.

"The routing network is adapted to selectively send messages to the nodes in the network" via a gateway. '722 Patent, 3:21-27. In an embodiment, updates "from an input source are assigned to one or more of N categories" and "the nodes are assigned to one or more of M types, and mappings are created between message categories and node types. Each gateway keeps track of these mappings" (e.g., category 1 data sent to type 1 nodes). *Id.* at 3:39-46; *see also* 18:24-51, Fig.

9. One way the '722 patent teaches assigning messages to categories "is to assign all message from a given input source [(e.g., information/content provider or its server] into a certain category." *Id.* at 18:52-53, 6:22-29. "When a gateway receives messages from input sources, the gateway identifies the categories of the messages and routes the messages to the nodes of the types to which the categories are mapped," and the nodes send the update messages to registered client devices. *Id.* at 3:43-46, 18:66-19:5.

C. Overview of the '785 Patent

The '785 patent claims a "virtual network manager [or system]" that includes several components: (1) a computing device or "network interface configured for data communication," (2) a "register module configured to register devices in a virtual network," (3) a router or route director, and (4) a "DNS server configured to receive a DNS request" from a first (source) device and return three addresses to the source, including "[(i)] a network address associated with a network route director, [(ii)] a private network address associated with a second [destination] device in the virtual network, and [(iii)] a virtual network address associated with the second [destination] device." '785 Patent, 36:37-56 (claim 30).

Representative claim 30 is below:

> [30.0] A virtual network manager, comprising:
>
> [30.1] a network interface configured for data communication via a virtual network that is defined by a domain name having an associated public network address;
>
> [30.2] a memory and a processor to implement a register module configured to register devices in a virtual network, the register module further configured to:
>
> [30.3] receive a registration request from an agent associated with a device;
>
> [30.4] distribute a virtual network address to the device when

> the device is registered in the virtual network, the device being identified to other devices in the virtual network by the virtual network address; and
>
> [30.5] a DNS server for the virtual network, the DNS server configured to receive a DNS request from a first device in the virtual network, and return a network address associated with a network route director, a private network address associated with a second device in the virtual network, and a virtual network address associated with the second device.

The '785 patent identifies the so-called IP address depletion problem as a challenge it was trying to solve to "enable the rapid creation of secure means that allows local and remote specified entities [computer devices] to communicate and collaborate from any location via a standard Internet connection." '785 Patent, 1:32-37. The '785 patent indicates that, due to its own popularity, the Internet is "running out of [IP] addresses" that can be assigned to each device, thereby limiting the ability for devices in private networks, for example, to communicate or be accessible via the Internet. *Id.* at 1:50-55. The '785 patent also purports to address the "mobility problem," involving devices being assigned constantly changing/dynamic IP addresses, which other devices on the Internet wishing to communicate with them might find difficult to track. *Id.* at 5:10-45. The '785 patent thus describes "a need for a system that provides for local and remote entities to communicate and collaborate using the Internet, [a system that] can work with existing NAT [(Network Address Translation)] devices and firewalls, and allows for devices to move to different physical networks." *Id.* at 2:45-48.

In the "Description of the Related Art" section, the '785 patent admits that NAT solutions where well known the art, and used to transparently translated between private and public IP address:

> One popular solution to the depleting address problem is [NAT]. This concept includes predefining a number of network addresses to be private addresses and public addresses. Public addresses are unique addresses that should only be used by one entity having access to the Internet. That is, no two entities on the Internet should have the same public address. Private addresses are not unique and are typically used for entities not having direct access to the Internet. Private addresses are used in private networks. Private addresses can be used by more than one organization or network. NAT assumes that all of the machines on a network will not need to access the Internet at all times. Therefore, there is no need for each machine to have a public address. A local network can function with a small number of one or more public addresses assigned to a NAT device. The remainder of the machines on the network will be assigned private addresses. Since entities on the network have private addresses, the network is considered to be a private network.

'785 Patent, 1:56-2:7.

One embodiment discloses a "secure Virtual Community Network or ('VCN')," described as "a private dynamic network which acts a private LAN for computing devices coupled to public networks or private networks." '785 Patent, 8:66- 9:5.

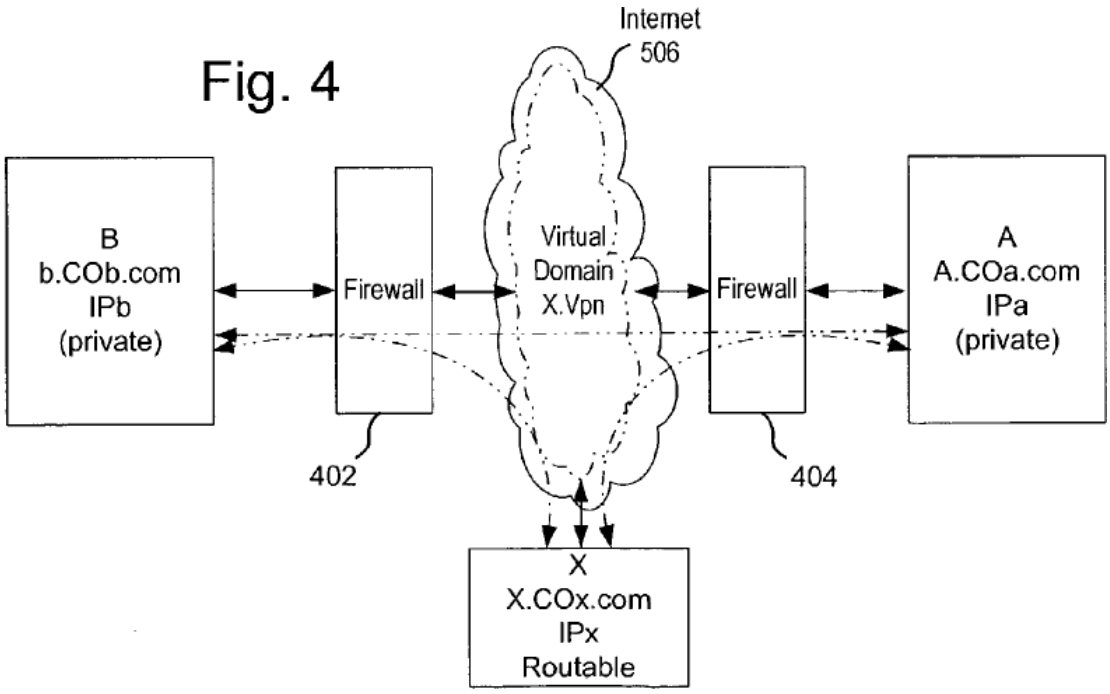


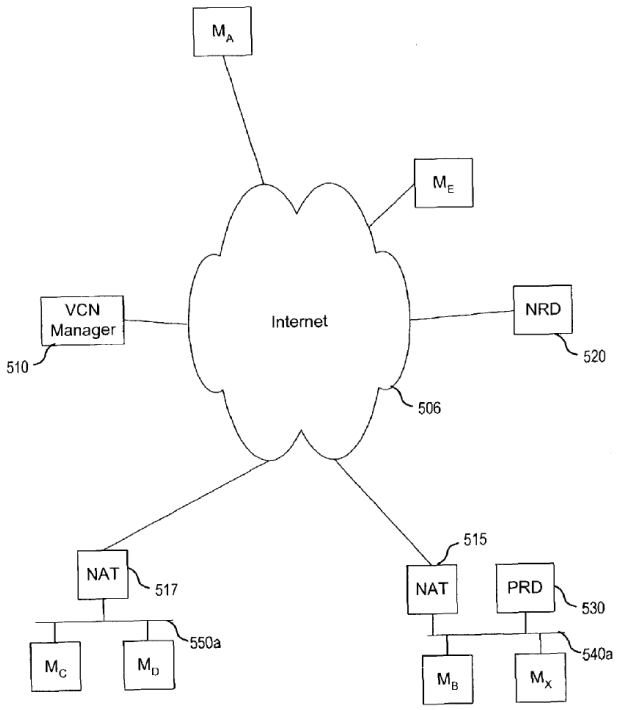
Figure 4 shows an exemplary VCN, which comprises a "computer or device B (host name—b.Cob.com) in a first private domain and computer or device A (host name—a.Coa.com) in a second private domain, both of which are coupled to the Internet by firewall devices 402, 404." *Id.* at 9:9-13. The firewall devices are NAT devices. *Id.* at 9:13-15. Moreover, "[c]omputer or device X is coupled directly to the Internet and has a public IP address." *Id.* at 9:16-17. "Devices A, B, and X can join the VCN, leave the VCN, or allows other devices in the VCN to communicate with them." *Id.* at 9:18-23. "All members of the VCN are accessible as if they were part of a physical local network" (e.g. all in the same private network)—allowing direct communication between devices within the virtual domain. *Id.* at 9:22-27.

The '785 patent states that "hardware architecture for the machines, servers, or other devices used to implement the present invention should be well understood to one of ordinary skill in the art," and outlines standard, readily available hardware/software components, including "one or more processors, a memory, a mass storage device, a portable storage device, a first network interface, a second network interface, and I/O devices, in communication with each other." '785

Patent, 9:42-48; *see id.* 9:48-54. The patent also states the network interface can include or be connected to a firewall. *Id.* at 9:57-60. The network interfaces are inside a computer, and when the computer also functions as a "router," it "includes two or more network interfaces;" *h*owever, "in other embodiments, the computer may include only one network interface." *Id.* at 9:54-56.

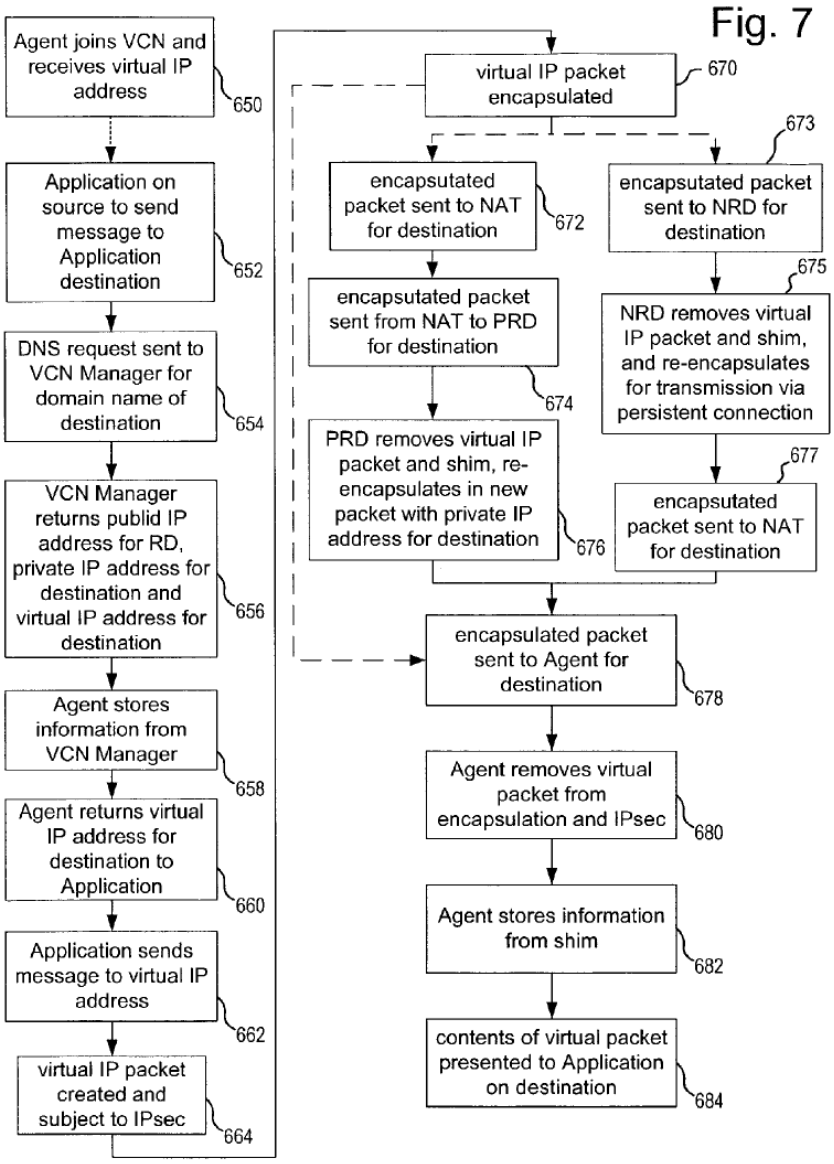
Another example of the system is illustrated in Figure 5A.

Fig. 5a



The VCN includes a VCN Manager 510, a Network Route Director 520 and/or Private Route Director 530, and a number of user devices (e.g., M_C, M_X, M_A etc.). .EX 1001, 10:24-27. "The VCN Manager 510 is a central server or server cluster providing management, connection, and security services for the VCN." ' 785 Patent, 10:66-11:1. The "various user machines, Mn, are coupled to the Internet 506, as well as other devices." '785 Patent, 10:17-20. "In order to establish or participate in communication within the VCN, a device must register with and join the VCN, thereby becoming a member." To join, "[t]he Agent will send a message to the VCN Manager 510

and receive a virtual IP address in return." *Id.* at 14:10-15. This is shown in step 650 of Figure 7. *Id.*



The '785 patent describes the virtual IP or network address as preferably another private network address that is chosen in a well-defined set of addresses (virtual address realm) such that each virtual network address occurs only once across the virtual network and thus is a unique identifier of the device within the network address. *Id.* at 12:50-54 ("each member is assigned a virtual IP address unique in the context of the virtual community and that will not conflict with

any local addresses on its LAN or any public addresses on the Internet."); *Id.* at 12:48-50 ("In general, when choosing virtual addresses it is best to choose addresses that will not be routable to the member."). To that end, the '785 patent describes this virtual network address in broad terms, adding that it can be a string, or an address compliant with either IPv4 or IPv6 standards. *Id.* at 12:40-44 ("Assigning a virtual address to each peer designated by a DNS string solves the problem of ambiguous endpoints ...These virtual addresses may be any legal IPv4 addresses.") As shown in Figure 7, a new member device wanting to send a message to another device in the VCN "will initiate a DNS request," which is then sent to the VCN Manager 510, as shown in step 654. *Id.* at 14:32-34. In response, the "VCN Manager 510 returns the public address of the Route Director for the destination, the private address for the destination device, and a virtual IP address for the destination device," as shown in Step 656. *Id.* at 14:34-38.[2]

This information is then stored in a table, or any other data structure" for creating a virtual IP packet to be forwarded to the destination address. '785 Patent, 14:48-15:15. When the destination wishes to respond to the source, or the sources wishes to send additional information, the process of Figure 7 can be repeated, although the steps of performing the DNS request do not need to be performed again, since the information between the two devices has already been stored by the devices. *Id.* at 15:58-67.

D. Overview of the '326 Patent

The '326 Patent concerns "a system and method for high data rate multi-channel wireless communications" which reuses "existing single channel radio designs" to "achieve very high data rate expanded bandwidth (wide band) WLAN operations." *See* Dkt#1 ¶ 15. Particularly, the '326

[2] The 785 specification notes that the three addresses are not always returned, for instance, in the case where "the destination machine has a public address (and, thus does not use a routing director) then step 656 will only include returning the public IP address for the destination." '785 Patent, 14:40-44.

patent seeks to provide "[a] method for increasing data rates and data throughput in a network" by "partially filling the frequency gap between the first channel and the second channel by adding one or more data subcarriers into the frequency gap such that the one or more guard bands are at least partially filled with at least some of the one or more data subcarriers," "combining the first channel and the second channel using channel bonding with orthogonal frequency division multiplexing," and "transmitting data subcarriers occupying the first channel, the second channel, and the frequency gap in parallel to a receiver." *Id.* at 1:1-24. The patent contemplates "a way to achieve very high data rate expanded bandwidth (wide band) WLAN operations reusing existing single channel radio designs" that "provides a dual-channel form of operation for greater flexibility and performance across multiple platforms." Dkt#1-4 at 7:59-63.

See Figure 4 of the '326 Patent below:

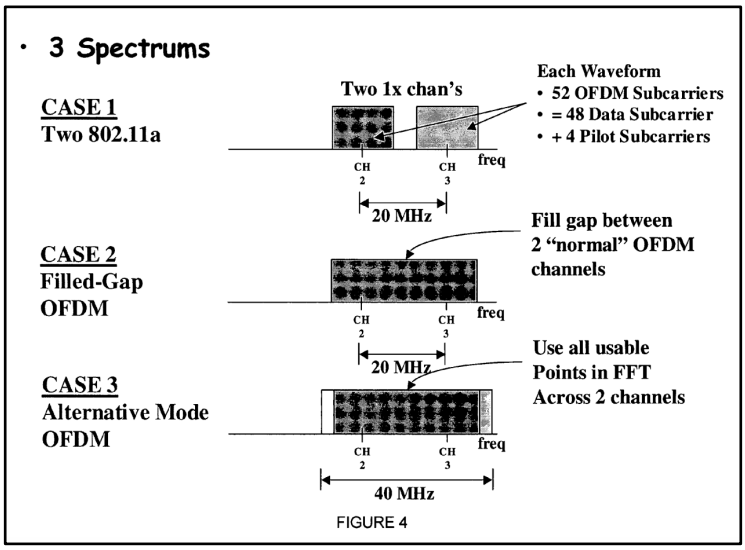


FIGURE 4

E. Overview of the '469 Patent

The '469 Patent concerns installing a satellite distributed Internet hotspot in a "remote location" or "rural area" that "experience[s] high volume transient traffic, such as rest area, restaurants, truck stops, rural hotels, conference centers, motels and state park lodges." '469 Patent at Abstract, 3:52-55. The specification and claims recite placing a "satellite dish 14 for

communicating with the satellite, a subscriber access unit 16 operatively coupled to the satellite dish 14 and a router 18 operatively coupled to the subscriber access unit"—"at the rural location," meaning "anywhere there is 120 volt electricity available or access to the sun for a solar panel and enough space to house the transceiver and mount a satellite dish." *Id*. at 1:39-41, 4:21-25, 4:18-28.

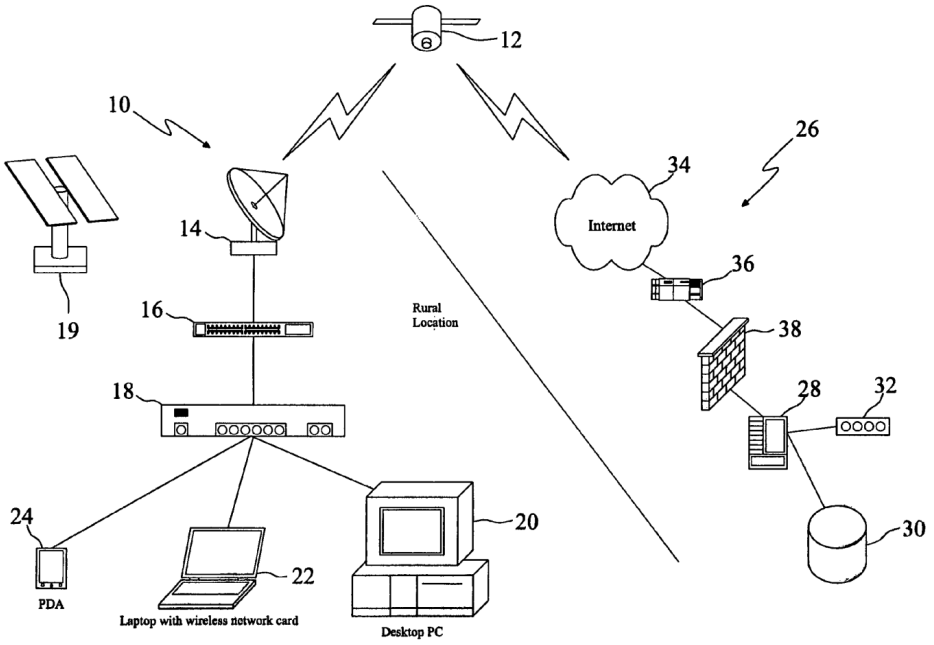


FIG.1

As shown in Figure 1 of the '469 Patent, the invention contemplates connection to the hotspot by "user web-enabled devices, such as PC 20, laptop computer 22 and PDA 24" (*id.* at 4:29-30), upon which

> they are routed through the 'Subscriber Access Unit.' This unit, utilizing radius client, automatically forces the user to first logon via satellite through a third party commercial site that houses the radius server. Internet access is granted upon activation of a valid credit card, or existing valid account.

'469 Patent at 4:2-7; *id.* at 4:32-34 ("The user can access the Hotspot using his own web-ready device to acquire a dynamic IP address").

F. Overview of the '582 Patent

The '582 Patent relates to load balancing using multiple computer processors. '582 Patent, 1:14-17. Particularly, '582 Patent deals with "a way to split the performance of a given task among a plurality of processing units which can all access, directly or indirectly, the input data and the devices on which the output data is to be stored." *Id.* Further, "[t]he principal object of the present invention is to enable the decomposition of a certain type of linear processes that currently use a single computer, into equivalent parallel processes that can efficiently use any number of potentially heterogeneous computers, taking the available capacity of each of these computers into account while optimizing execution." *Id*. at 1:48-54. See FIG. 1 of the '582 Patent, below.

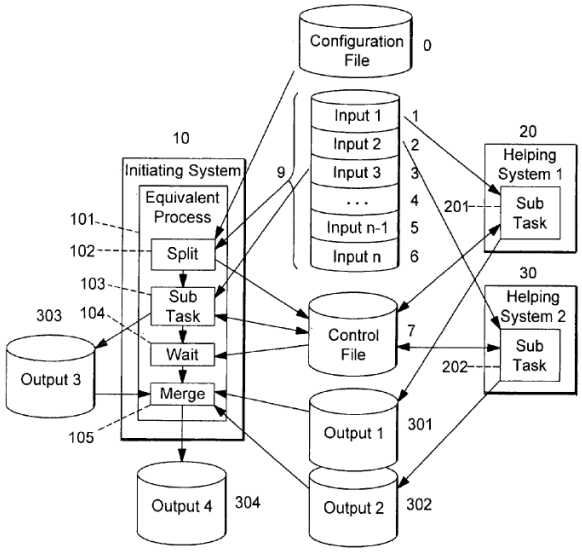


Fig. 1

* * *

Each of the concepts discussed above, as well as many others, were widely known by the alleged invention dates of the Asserted Patents. Accordingly, the Asserted Patents are nothing

more than a recitation of knowledge that was well-known in the industry, and are, at best, a recitation of well-known features and functionalities that took no more than an ordinary level of skill in the art to arrange in a known manner using conventional techniques.

III. GROUNDS OF INVALIDITY UNDER 35 U.S.C. § 101

Pursuant to the Court's OGP, Southwest identifies the following Asserted Claims of the Asserted Patents as ineligible under 35 U.S.C. § 101. In addition to the specific discussions below, the specification and prosecution history for each asserted patent reflect statements and admissions concerning the state of the art at the time of the purported inventions, and the prior art charted and discussed in these contentions for purposes of invalidity under 35 U.S.C. § 103 is relevant for consideration regarding the state of the art at the time of the purported inventions.

A. U.S. Patent No. 8,332,844

1. State of the Art Relevant to the '844 Patent

By the earliest priority date of December 30, 2004, systems administrators and infrastructure engineers working in distributed computing environments had access to a range of well-understood, conventional technologies for provisioning and managing operating system images across multiple compute nodes. These technologies were not only theoretically described but actively implemented and in widespread use across educational institutions, research laboratories, enterprise networks, and government systems. Understanding this state of the art is critical to evaluating the Asserted Claims of the '844 Patent, as it establishes that the techniques described in the patent were already conventional, mature, and widely implemented in the relevant field.

Distributed system architectures at the time were designed to deliver consistent runtime environments to groups of machines—whether stateless thin clients, HPC nodes, or clustered servers—by employing shared root filesystems, overlay filesystems, and caching mechanisms.

These established practices optimized storage, consistency, and performance and were actively documented and deployed in real-world systems.

a. Distributed Image Management in 2004

By December 2004, several foundational techniques in distributed image and storage management were widely recognized, well understood, and routinely implemented across both academic and commercial systems. These included the use of snapshots, copy-on-write (CoW) mechanisms, merging delta changes with base images, shared base images with writable overlays, and efficient virtual machine migration strategies.

One of the most prevalent techniques was snapshotting using Copy-on-Write. Storage systems like NetApp's Write Anywhere File Layout (WAFL) architecture leveraged CoW to create space-efficient, consistent, point-in-time copies of file systems. WAFL enabled the creation of read-only snapshots and writable clones that separated static base data from dynamic changes without duplicating the entire file system. The concept was not limited to NetApp; as early as 1992, the Episode File System integrated snapshots and file versioning with CoW in distributed environments supported by the Distributed Computing Environment (DCE). In the virtualization space, VMware's early GSX and ESX platforms also relied on CoW to manage snapshots of virtual machines, tracking changes made after a snapshot was taken while preserving the original disk image.

Alongside snapshotting, it was routine to merge snapshots with base images to present a unified view of the file system or virtual machine state. For example, VMware systems combined base virtual disk images with delta files to construct a current VM state, a practice commonly used by system administrators by the early 2000s. Similarly, NetApp technologies described in patents such as U.S. Patents 5,819,292 and 6,604,118 disclosed mechanisms for merging base data with CoW-based snapshots to provide a consistent and current data view.

In networked and high-performance computing environments, the use of shared read-only base images with writable overlays was equally well established. Systems such as Sun Microsystems' diskless workstations and the Linux Terminal Server Project (LTSP) enabled multiple clients to boot from a single centralized read-only root image while maintaining isolated writable overlays. Academic and research environments employing Beowulf clusters or Rocks Cluster Toolkit adopted similar strategies, allowing efficient management of large numbers of nodes while minimizing image duplication. NetApp further enabled writable clones of read-only snapshots, as disclosed in U.S. Patent 7,334,095.

The migration and management of virtual machines—including their associated image states and snapshots—was also well understood by 2004. Research by Sapuntzakis *et al.* at Stanford University demonstrated the migration of virtual machines using CoW disks and demand paging to efficiently capture and transfer VM state across systems. These techniques mirrored the commercial practices already in use by companies like VMware and reflected broader industry familiarity with image management in virtualized environments.

b. Real-World Implementations Demonstrating the State of the Art

Several systems operational before or during 2004 embodied the above techniques, demonstrating that they were not only theoretically known but also widely implemented and recognized in practice.

- NetApp's WAFL File System exemplified the state of the art with its CoW-based snapshot architecture. WAFL enabled the creation of read-only snapshots, writable clones, and consistent file system views without duplicating the base image. This architecture was thoroughly documented in both U.S. Patents 5,819,292 and 7,334,095, as well as in technical literature such as "File System Design for an NFS File Server Appliance" by Dave Hitz et al.

- VMware's GSX and ESX server platforms provided commercial-grade virtual machine management, including snapshotting, delta image tracking, and the merging of these deltas with base images. This allowed users to roll back or move between states while minimizing disk usage.
- Sun Microsystems' diskless workstations leveraged centralized read-only images to boot multiple clients across a network, each maintaining private writable layers. These overlays captured client-specific changes while preserving the shared base image intact.
- The Episode File System, developed in the early 1990s, brought snapshotting and version control to distributed file systems using CoW. This system efficiently tracked and stored changes to files across versions in a distributed environment.
- Petal, a distributed virtual disk project from HP Labs, provided dynamic, fault-tolerant storage with support for snapshot creation and management using CoW techniques. Petal's architecture demonstrated the application of distributed block storage principles in combination with image versioning and migration features.

Taken together, these examples show that by December 2004, the claimed techniques for managing images, snapshots, and overlays in distributed systems were neither novel nor inventive—they were standard practice in both the commercial and academic realms.

c. Conclusion:

By the earliest priority date of December 30, 2004, the concepts underlying the Asserted Claims of the '844 patent were well understood, routine, and conventional within the field of distributed system image management. The use of snapshots, CoW, merging base images with

snapshots, and maintaining shared root images with per-client writable overlays were well documented, publicly available, and widely deployed.

These practices were integral to well-established systems such as NetApp WAFL, VMware virtualization environments, Sun Microsystems diskless workstations, and cluster management solutions. Because the techniques described by the Asserted Claims of the '844 Patent were routinely implemented and broadly accepted by December 2004, the Asserted Claims of the '844 patent are both conventional and rendered obvious by the state of the art.

2. The Asserted Claims of the '844 Patent Fail at *Alice* Steps 1 and 2

The '844 Patent, titled "Root Image Caching and Indexing for Block-Level Distributed Application Management," claims methods for organizing and storing information. Dkt#1-1. Claim 7—the only independent claim of the Asserted Claims for the '844 Patent—recites:

> 7. A method for **providing data to a plurality of compute nodes**, comprising:
>
> **storing blocks of a root image** of said compute nodes **on a first storage unit**;
>
> **storing leaf images** for respective compute nodes **on respective second storage units**, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and
>
> **caching blocks of said root image** that have been accessed by at least one of said compute nodes **in a cache memory**.

The specification admits that the claimed components and methods—clustered computing, root-leaf storage systems, and caching mechanisms—were conventional. For example:

- Clustered computing and distributed file systems were established practices (Dkt#1-1 at 1:31–45);
- Root-leaf storage systems were a common method for creating boot images "on the

fly" (*Id.* at 2:14–24); and

- Caching mechanisms, including least-recently-used (LRU) algorithms, were known (*Id.* at 6:38–7:17).

Dependent claim 11 simply adds a generic "merging" step to claim 7—joining the blocks of the root and leaf images to "create cohesive respective application environments." Nonetheless, at their core, claims 7 and 11 are directed to the abstract idea of organizing and storing data using a hierarchical structure of a shared base and individualized overlays. The '844 Patent relies entirely on general-purpose computing devices with generic storage media (*Id.* at 4:27–56), and unclaimed components, such as "Union Block Devices" (UBDs), described as low-level drivers with no disclosed novel functionality (*Id.* at 5:19–64). The Asserted Claims of the '844 Patent likewise employ broad functional language like "providing," "storing," and "caching," without specifying any improved algorithms, hardware, or software. Thus, the Asserted Claims of the '844 Patent merely recite an abstract idea for organizing and storing data without any inventive concept, thereby failing at both steps of the *Alice* test.

a. The Asserted Claims of the '844 Patent recite an abstract idea for organizing and storing data.

The Asserted Claims of the '844 Patent are directed to the abstract idea of organizing and storing information using a "root-leaf" structure. Although applied to a computer network, the method steps—storing root and leaf images and caching accessed blocks—mirror basic data collection and organization practices, analogous to longstanding human activities. The patent attempts to claim the concept of storing some information in a central repository for permanent storage and common use (a root image), storing other non-identical data locally for faster retrieval and personal use (leaf images), and keeping recently used items temporarily on hand (cached). Such abstract ideas for how to organize and store data on computer systems are one of "the 'basic

tools of scientific and technological work' that are free to all men and reserved exclusively to none." *Synopsys, Inc. v. Mentor Graphics Corp.*, 839 F.3d 1138, 1146 (Fed. Cir. 2016) (quoting *Alice*, 573 U.S. at 216). Courts have consistently held such practices abstract. *See Intellectual Ventures I LLC v. Erie Indem. Co.*, 850 F.3d 1315, 1327 (Fed. Cir. 2017) (organizing and accessing data held abstract); *Content Extraction & Transmission LLC v. Wells Fargo Bank, Nat'l Ass'n*, 776 F.3d 1343, 1347 (Fed. Cir. 2014) (data collection and storage held abstract); *Kaavo Inc. v. Amazon.com Inc.*, 323 F. Supp. 3d 630, 641 (D. Del. 2018) ("setting up and managing a cloud computing environment" held abstract); *Versata Software, Inc. v. NetBrain Techs., Inc.*, No. 13-676-LPS-CJB, 2015 WL 5768938, at *7 (D. Del. Sept. 30, 2015) ("representing information in a hierarchy amounts to an abstract idea.").

The specification confirms that the described components and techniques, including root-leaf storage systems and caching mechanisms, were well understood at the time of the invention. Dkt#1-1 at 1:31–45, 6:37–7:5. The Asserted Claims of the '844 Patent provide no specifics on storage structures/hardware, caching algorithms/techniques, or other technical improvements, relying instead on functional language that describes high-level results without specific implementation details. *See ChargePoint, Inc. v. SemaConnect, Inc.*, 920 F.3d 759, 769 (Fed. Cir. 2019) ("Even a specification full of technical details about a physical invention may nonetheless conclude with claims that claim nothing more than the broad law or abstract idea underlying the claims.").

Claim 7's scant details about what data gets saved where are inherently abstract. *See Content Extraction*, 776 F.3d at 1347 ("The concept of data collection, recognition, and storage is undisputedly well-known. Indeed, humans have always performed these functions."). "This type of activity, *i.e.*, organizing and accessing [data], includes longstanding conduct that existed well

before the advent of computers and the Internet." *Erie Indem.*, 850 F.3d at 1327. Claim 7 includes no purportedly novel technical details for cluster computing—just an abstract idea for how to organize and store data. In the end, the claim is little different from a librarian storing different types of books and papers in different sections of a library. Claim 7 therefore fails *Alice* Step 1 because it recites a conventional root-leaf storage system using a high-level abstract concept to organize and store data without any specific technological improvement recited in the claim. Claim 11 fares no better; it merely adds a generic "merging" step—joining the blocks of the root and leaf images to "create cohesive respective application environments"—without describing *how* that merging is performed or any technical rules guiding it. Claim 11 therefore provides no new or inventive mechanism for merging, just the result.

b. The Asserted Claims of the '844 Patent lack an inventive concept.

Under Step 2 of the *Alice* framework, a claim must include an "inventive concept" sufficient to transform an abstract idea into patent-eligible subject matter. *Alice*, 573 U.S. at 217. To qualify, the claim must add something "significantly more" than the abstract idea itself, such as an improvement to the functioning of a computer or a specific application of technology. *See Erie Indem.*, 850 F.3d at 1328. "If a claim's only 'inventive concept' is the application of an abstract idea using conventional and well-understood techniques, the claim has not been transformed into a patent-eligible application of an abstract idea." *BSG Tech LLC v. Buyseasons, Inc.*, 899 F.3d 1281, 1290–91 (Fed. Cir. 2018). "The written description is particularly useful in determining what is well-known or conventional." *Intellectual Ventures I LLC v. Symantec Corp.*, 838 F.3d 1307, 1317 (Fed. Cir. 2016). Moreover, "a claimed invention's use of the ineligible concept to which it is directed cannot supply the inventive concept that renders the invention 'significantly more' than that ineligible concept." *BSG Tech*, 899 F.3d at 1290.

The Asserted Claims of the '844 Patent likewise fail Step 2, in part, because "these claims use generic computers to perform generic computer functions." *Symantec*, 838 F.3d at 1315. The specification concedes that the claimed methods are standard, including storing root images on a first storage unit, leaf images on second storage units, and caching accessed blocks. Dkt#1-1 at 2:14–21, 6:38–67. Thus, even if the specification discloses ideas that could be applied to useful ends, claims 7 and 11 "contain[] no restriction on how the result is accomplished." *Symantec*, at 1316.

The claimed steps are described at a high level of generality, with no new algorithms, configurations, or technological improvements. Generic terms like "providing," "storing," and "caching" describe desired outcomes without specifying any "inventive concept" for how these tasks are performed. *See BSG Tech*, 899 F.3d at 1290-91 ("If a claim's only 'inventive concept' is the application of an abstract idea using conventional and well-understood techniques, the claim has not been transformed into a patent-eligible application of an abstract idea"); *Elec. Power Grp., LLC v. Alstom S.A.*, 830 F.3d 1350, 1356 (Fed. Cir. 2016) (drawing the "distinction between ends sought and particular means of achieving them, between desired results (functions) and particular ways of achieving (performing) them").

Breaking down each element of the Asserted Claims of the '844 Patent reveals how each element is conventional:

Claim 7:

- **"storing blocks of a root image of said compute nodes on a first storage unit"**

  This is conventional. Shared read-only root images were routinely stored on central storage devices, as used in Sun's diskless workstations and LTSP environments. (*See* Hitz et al.; NetApp WAFL).

- **"storing leaf images for respective compute nodes on respective second storage units…"**

  This element is likewise conventional. Systems such as VMware and Episode stored deltas or overlays for individual machines using CoW. These deltas excluded unchanged base data, which was referenced from the shared root image.

- **"caching blocks of said root image that have been accessed…"**

  This element is likewise conventional. Caching mechanisms, including block-level caching and LRU algorithms, were widely used by 2004 in both operating systems and storage appliances (*see* NetApp; LTSP).

Claim 11:

- **"merging said blocks of the root image and said leaf images…"**

  The element recited by Claim 11 is similarly conventional and routine. The merging recited by Claim 11 had been widely implemented in systems such as VMware's and NetApp's platforms.

Courts have invalidated analogous claims. In *Symantec Corp.*, 838 F.3d 1307, the Federal Circuit invalidated claims for managing email messages using conventional computer components. Similarly, the court in *Content Extraction*, 776 F.3d 1343, held claims for data recognition and storage invalid because they relied on routine computing methods. The Asserted Claims of the '844 Patent fall into the same category, as they merely recite routine computing techniques and hardware.

The Asserted Claims of the '844 Patent offer no inventive step, combining conventional techniques predictably to achieve abstract goals. They rely on generic computing elements and

routine data management practices, failing *Alice* Step 2. Thus, The Asserted Claims of the '844 Patent are ineligible under §101.

B. U.S. Patent No. 8,407,722

1. State of the Art Relevant to the '722 Patent

At the time of the purported invention, claimed to have a priority date of December 18, 2000, the field of asynchronous messaging, client registration, and dynamic content routing was well established. Numerous standards, protocols, and technologies had already addressed the challenges of efficiently distributing data updates in networked environments, particularly where client devices needed to receive specific information based on prior registration.

The alleged innovation in the '722 Patent centers around organizing and transmitting data through a method that involves client registration and message routing within a dynamic network architecture. However, as demonstrated below, these concepts were not novel or unconventional as of December 2000.

a. Networking Standards and Protocols (Pre-2000)

One of the foundational concepts purportedly improved by the '722 Patent is asynchronous messaging, where data updates are sent without requiring an immediate response from the client. Asynchronous data routing was not a novel concept at the time of the alleged invention; it had been widely implemented in established systems and protocols well before the priority date.

One of the earliest implementations of dynamic data transmission is IP Multicast, standardized as RFC 1112 in 1989. IP Multicast allows sending a single data transmission to multiple recipients who have joined a specific multicast group. This approach significantly reduced network traffic by ensuring that only registered devices received the transmitted data. The ability to direct data only to interested recipients mirrors the concept of client registration for data updates, as claimed in the '722 Patent.

Similarly, CORBA Event Services, introduced in the late 1990s, provided asynchronous communication between distributed objects. These services allowed clients to register for specific events or updates, forming a fundamental part of distributed application architectures. CORBA's event-driven model effectively addressed the need for targeted data distribution and client registration in networked systems, predating the patent's purported improvements.

Another significant standard, the Java Message Service (JMS), was introduced in 1998. JMS implemented both the publish-subscribe model and point-to-point messaging, allowing clients to subscribe to specific data topics and receive only the relevant updates. This capability directly addressed the problem of selective message delivery, which the '722 Patent later claimed to improve.

Likewise, TIBCO Rendezvous, available by 1998, utilized content-based addressing and routing to manage the efficient delivery of data in real-time applications. This system also incorporated message queuing and subscription mechanisms, showing that targeted data dissemination was a well-understood challenge long before the alleged invention.

b. Message Queuing and Client Registration Systems

During the 1990s, message queuing systems were already employing techniques for client registration and selective message distribution. IBM MQ Series, introduced in the early 1990s, provided robust message queuing with support for both publish-subscribe models and point-to-point communication. These systems enabled client applications to register for specific message types, thereby receiving only the updates of interest. This core functionality—registering for updates and receiving targeted messages—parallels the method claimed in the '722 Patent.

In a similar vein, Gryphon, a content-based routing system introduced in 1998, supported dynamic data routing and client subscriptions. It allowed clients to receive updates based on

content specifications, a method fundamentally similar to the dynamic message routing described in the '722 Patent.

Another early messaging protocol, Session Initiation Protocol (SIP), which was standardized in 1999, offered a subscription and notification mechanism for presence updates. Through SUBSCRIBE and NOTIFY functions, SIP enabled clients to receive real-time notifications based on their expressed interests, thus efficiently handling client-specific data routing.

c. Relevant Prior Art Patents (Pre-2000)

Several patents pre-dating the '722 Patent also addressed dynamic content routing and client registration.

- **US Patent 5,754,939 (Herz et al.)** (issued in **1998**) discloses methods for routing messages to recipients based on profile information. This technique inherently involves selectively delivering messages to clients who registered to receive them, thus predating the '722 Patent's claimed method of client-specific message updates.
- **US Patent 5,845,324 (White et al.)** (issued in **1998**) similarly addresses message routing in networked communication systems, emphasizing methods where client devices receive only the relevant data based on their registrations.
- **US Patent 5,919,247 (Van Hoff et al.)** (issued in **1999**) discusses distributed content update systems that involve client-server architectures to update registered clients. This system reflects the idea of routing updates only to those clients that have expressed a need for specific data, which the '722 Patent claims as an improvement.

d. Real-Time Data and Event Notification Systems

By the late 1990s, event notification systems were leveraging client subscriptions to deliver updates efficiently. For example, Event Notification Services from 1999 utilized client-specific subscriptions to notify registered users of changes or updates, mirroring the client registration and update routing claimed in the '722 Patent.

Another significant example from 1996 is the Dissemination-Based Information Systems research, which focused on efficiently delivering data to registered users. These systems implemented techniques for selectively transmitting updates, thereby optimizing network resources, much like the patent's method of reducing message traffic.

e. Industry Implementations (Pre-2000)

In practice, industries such as finance had long implemented real-time data systems that updated registered clients with new information, such as stock price changes. Middleware systems like IBM MQ and CORBA were commonly used in enterprise environments to route messages efficiently, leveraging client subscriptions and content-based routing well before the priority date of the '722 Patent.

f. Conclusion

The evidence shows that by December 2000, the concepts underlying the '722 Patent—specifically client registration, dynamic content routing, and efficient message dissemination—were well-established in the field. The alleged invention does not introduce a novel technological improvement but rather combines existing methods and standards in a predictable way. The claimed methods represent an incremental combination of known techniques rather than a groundbreaking departure from the established state of the art.

2. The Asserted Claims of the '722 Patent are Ineligible Under 35 U.S.C. § 101

The Asserted Claims (claims 14, 16, 17, and 18) of the '722 Patent recite the abstract idea of receiving information, identifying relevant content, registering interest, and disseminating updates to interested parties. This is a long-standing and fundamental practice in organizing and distributing information—not a technological improvement or specific technical solution. Because the claims implement an abstract idea using only generic networking and data-handling functions without reciting any technological improvement or unconventional hardware, the claims fail both steps of the *Alice* framework.

a. The Asserted Claims of the '722 Patent are Directed to an Abstract Idea

The Asserted Claims of the '722 Patent are directed to the abstract idea of receiving information, identifying relevant content, registering interest, and disseminating updates to interested parties. Claim 14—the only representative independent claim—describes a method for enabling dynamic updates to "live objects" displayed on a client device through a structured communication framework involving a source device, a routing network, and intermediary nodes. An input source sends a data representation to a client device via a routing network; this data includes at least one live object—an identifiable element capable of receiving updates. The client device recognizes the live object, determines its identifier, and registers with the routing network to receive updates about it. This registration includes connection information, which is stored at a node in the routing network. When the input source sends an update message for the live object, the message is routed through a gateway that determines its category and selects the appropriate node type to handle it. The node then identifies the registered client and forwards the message. Upon receipt, the client updates the relevant property of the live object.

Claim 16 adds to claim 14 that the client stores metadata related to the live object, and that the update message modifies this metadata. Claim 17 adds to claim 14 that the update message includes a message type, which the client uses to determine how to apply the update. Claim 18 expands on claim 14 by allowing the client to register for multiple live objects and receive updates for each.

What the Asserted Claims of the '722 Patent effectively describe is a rules-based information forwarding mechanism, based on identifiers and interest registration—a framework that has existed in mailrooms, subscription services, and newsrooms long before the advent of digital networks. Its implementation using computers and a "routing network" merely automates this longstanding human practice.

Stripped of jargon, the Asserted Claims of the '722 Patent describe a familiar, abstract pattern: sending information, having a recipient identify what is relevant, and distributing future updates to those who have expressed interest. This pattern is not new or technological. It is, for example, identical in substance to how a traditional newsroom operates. A journalist (the "input source") provides a summary of several developing stories (the "data representation") to a central bulletin board in the newsroom. Each story on the board is a "live object" in the sense that it may be updated. Editors (the "client devices") review the bulletin board and mark their interest in particular stories. When updates come in from the field, a receptionist (the "gateway device") categorizes the update and determines which editor (the "node") should receive it. The update is then delivered to the relevant editor, who revises the draft story accordingly.

This real-world newsroom example illustrates that the Asserted Claims of the '722 Patent merely recite an abstract method of organizing and delivering information—a practice that has been implemented manually for decades. The fact that the '722 Patent describes this process in the

context of a computing environment does not render it any less abstract. Claims directed to abstract frameworks for information delivery—like updating content in response to user interaction or preference—have repeatedly been held patent-ineligible. *See, e.g.*, *Broadband iTV, Inc. v. Amazon.com, Inc.*, No. 6:20-CV-00921-ADA, 2022 WL 4703425, at *11 (W.D. Tex. Sept. 30, 2022) (J. Albright), *aff'd,* 113 F.4th 1359 (Fed. Cir. 2024); *see also Broadband iTV, Inc. v. Oceanic Time Warner Cable, LLC*, 135 F. Supp. 3d 1175, 1178 (D. Haw. 2015) (claims directed to abstract idea of "using the same hierarchical ordering based on metadata to facilitate the display and locating of video content" ineligible under § 101). Because the Asserted Claims of the '722 Patent focus on the result of an implementing an abstract idea—delivering updated content to interested recipients—rather than a specific technological means of achieving that result, they fail at *Alice* step one.

The specification attempts to present these steps as an improvement over prior art by claiming to enhance network efficiency through dynamic client registration and message routing. It asserts that the use of node specialization and dynamic mapping reduces unnecessary message traffic. The routing network purportedly eliminates the need for the input source to track which clients are displaying the live objects by maintaining a registry of registered clients. The system is described as being useful for applications where content on client devices must be updated dynamically based on changes in data, such as stock prices or sports scores. However, the specification itself reveals that the claimed methods are implemented using conventional networking techniques. The routing network, which consists of nodes and gateways, is built using generic computing components, including single-processor computer systems running common operating systems like Linux. The specification indicates that the system utilizes existing messaging protocols, such as HTTP, and known client-server architectures. Furthermore, the

patent acknowledges that the basic concepts of client registration and update handling were already present in existing methods that included client-driven approaches, applets, and server-driven updates using TCP/IP connections.

b. The Asserted Claims of the '722 Patent Do Not Recite an Inventive Concept

Step 2 of *Alice* cannot save the Asserted Claims of the '722 Patent from ineligibility under 35 U.S.C. § 101. The Asserted Claims of the '722 Patent do no more than implement the abstract idea of selectively distributing content updates to interested recipients using generic computing components performing routine, conventional functions. The purported improvement—routing update messages based on client registration—does not introduce any new computing technology or method of enhancing network efficiency. Instead, it merely applies well-known client-server communication techniques to the problem of distributing data updates. The specification itself discloses that the system utilizes conventional networking components and known data transmission protocols without any significant modification.

Claim 14 recites basic components: an "input source," a "client device," a "routing network," "nodes," and "gateways." But none of these are claimed in a specific or inventive configuration. The specification describes the input source as a web server or dynamic content provider, the client device as a browser-enabled device such as a PC or mobile phone, and the routing network as an internet-based message delivery structure. The "gateway" and "node" elements, while given roles in the message flow, are described in general terms that mirror conventional content delivery mechanisms—such as proxies, relays, load balancers, or intermediate servers.

There is no technical detail in the claim about how messages are classified, how registration is implemented, or how update routing is performed. The claim merely recites the expected result:

that a client receives an update for a live object it previously registered for. Courts have routinely held that such claims, which rely on routine network behavior and leave implementation to the practitioner, do not supply an inventive concept. *See, e.g.*, *Two-Way Media Ltd. v. Comcast Cable Commc'ns, LLC*, 874 F.3d 1329, 1339 (Fed. Cir. 2017) ("The claim uses a conventional ordering of steps—first processing the data, then routing it, controlling it, and monitoring its reception—with conventional technology to achieve its desired result.").

The specification itself confirms that the invention is designed to operate on existing infrastructure and leverages known technologies. Client devices are said to use standard web browsers with existing support for Java, JavaScript, or plug-ins. Update messages are transmitted over HTTP and are parsed using client-side scripts. The routing network is implemented using conventional clusters, nodes, and gateways, and messages are routed using category/type mappings or registry lookups—well-understood design patterns in networked systems. The so-called "live objects" are simply HTML elements or other page components that can be identified by an ID and updated via DOM manipulation or script execution, which common techniques in web development since at least the late 1990s. The use of an "activation module" to register interest in updates is just another name for client-side code that subscribes to a push service—an architecture long practiced in content syndication and notification systems. Nothing in Claim 14 requires, or even enables, a novel technical implementation of these mechanisms.

None of the limitations in claims 16, 17, or 18 inject any inventive concept into the '722 Patent:

- **Claim 16**'s mention of metadata storage and update is a routine feature of client-side state tracking. Web applications and software clients have long-maintained associated metadata (e.g., timestamps, tags, formatting) alongside core content.

- **Claim 17** introduces a message type field used to determine how to process an update. But conditional logic based on message types is a staple of basic programming and messaging protocols. This is no more inventive than a "switch" statement tied to a message header.
- **Claim 18** expands registration to multiple live objects, which is a straightforward extension of the core idea. There is no novel mechanism for batch registration, grouping, or prioritization; the claim merely allows more than one subscription, a feature inherent in any scalable publish-subscribe system.

Thus, the dependent claims likewise apply well-known programming constructs and user-experience features without claiming how any of them are implemented in a novel or unconventional manner.

Furthermore, the combination of client registration and dynamic message routing was not an inventive concept at the time of the alleged invention. Technologies such as IP Multicast, Java Message Service (JMS), and CORBA Event Services were already implementing message routing based on client registration well before the priority date. The combination of these known elements does not produce unexpected results or provide a technological improvement. Here, the claimed method merely takes existing client registration and message routing techniques and applies them in a straightforward, predictable way.

The Asserted Claims of the '722 Patent therefore do not improve any specific computer functionality or solve any technical problem in a new way—they merely implement a conventional information delivery model using abstract functional language and general-purpose computing components. As such, the Asserted Claims of the '722 Patent fail *Alice* step two and are invalid under 35 U.S.C. § 101.

c. Conclusion

The Asserted Claims of the '722 Patent are invalid under 35 U.S.C. § 101 because they fail both prongs of the *Alice* framework. At step one, these claims are directed to the abstract idea of distributing updates about selected content to interested recipients—an organizing principle that has long been practiced by humans and implemented in conventional information systems. These claims describe this idea at a high level of generality, using functional language untethered to any specific improvement in technology or network design. At step two, the claims do not recite any inventive concept sufficient to transform this abstract idea into a patent-eligible application. The components are generic, the operations are conventional, and the implementation relies entirely on known computing techniques. These claims simply automate a familiar content delivery paradigm using routine architecture, offering no technical advancement over the prior art.

C. U.S. Patent No. 7,949,785

The Asserted Claims (claims 30, 35, and 37) of the '785 Patent are invalid under 35 U.S.C. § 101 because they are directed to an abstract idea and lack an inventive concept. The Asserted Claims of the '785 Patent recite a virtual network manager that facilitates secure communications within a virtual community—an abstract concept under *Alice* step 1. Further, the claims use only generic and conventional computing components and processes without introducing any specific technological improvements, and as such, the claims likewise fail to recite an inventive concept necessary to survive *Alice* step 2. Consequently, the Asserted Claims of the '785 Patent fail under *Alice* and are ineligible for patent protection under 35 U.S.C. § 101.

1. State of the Art in March 2003

By March 2003, secure communication and virtual networking were well-established fields, and numerous technologies were routinely used to address the challenges associated with secure data transmission over public networks. The '785 Patent, which claims a virtual network

manager for secure communication in a virtual community, builds on concepts and techniques that were widely known and used by March 2003. To fully understand the context of the invention, it is essential to examine the prevailing technologies and methods that a person of ordinary skill in the art (POSA) would have recognized as routine and conventional.

a. Secure Communication and Virtual Private Networks (VPNs)

One of the most common methods for secure communication over public networks in 2003 was the use of Virtual Private Networks (VPNs). VPNs were routinely deployed to create secure tunnels that allowed private network traffic to be transmitted over the public Internet. These tunnels were established using encryption protocols such as IPsec (Internet Protocol Security), which provided authentication, encryption, and data integrity for IP communications. IPsec was commonly used to create secure point-to-point connections, effectively encapsulating data within an encrypted packet that could be transmitted securely over the Internet.

Other VPN protocols included the Point-to-Point Tunneling Protocol (PPTP), which facilitated the creation of VPNs by encapsulating data within IP packets, and Secure Sockets Layer (SSL)/Transport Layer Security (TLS), which were widely used for web-based secure communication. VPN technology was an integral part of network security practices, with major companies like Cisco offering VPN client software and integrated VPN solutions as part of their networking equipment. The concept of encapsulating private network traffic within a secure channel was routine and had been implemented in numerous commercial and enterprise settings.

Further, the use of VPNs addressed the growing need for secure remote access, allowing employees to connect to a corporate network from offsite locations. By encrypting data and using secure tunneling, VPNs enabled organizations to maintain private communications even when using the public Internet. This approach was considered standard practice for enterprises looking to protect data from unauthorized access.

b. Network Address Translation (NAT) and Address Depletion

During this period, the Internet was facing a well-documented problem: the depletion of available IPv4 addresses. The rapid expansion of Internet-connected devices had exhausted the finite pool of unique public IP addresses. Network Address Translation (NAT) emerged as a practical and widely implemented solution to this problem. NAT allowed multiple devices within a private network to share a single public IP address when accessing the Internet. This was achieved by translating the private IP addresses to the public IP address of a NAT device (such as a router) and tracking the active connections using port numbers.

NAT was particularly beneficial for enterprises with large internal networks, as it significantly reduced the number of public IP addresses needed. NAT devices maintained tables mapping internal (private) addresses to external (public) ones, enabling inbound and outbound traffic to be correctly routed. This technique was so fundamental that most consumer and enterprise routers incorporated NAT functionality by default.

The concept of NAT traversal was also well understood by 2003. Because NAT inherently interfered with direct peer-to-peer communication, solutions like STUN (Session Traversal Utilities for NAT) and UPnP (Universal Plug and Play) were developed to allow devices behind NAT to establish direct connections. These techniques were vital for applications such as VoIP (Voice over IP) and real-time communications, which required consistent end-to-end connectivity despite NAT barriers.

c. Dynamic DNS (DDNS) and Mobility Solutions

Another challenge related to IP address management was the dynamic assignment of IP addresses to devices, especially in mobile or residential contexts. Dynamic DNS (DDNS) provided a solution by automatically updating DNS records when a device's IP address changed. This was crucial for maintaining connectivity to devices that did not have a fixed IP address. DDNS allowed

users to maintain a consistent hostname that would resolve to the device's current IP address, enabling continuous access regardless of network changes.

Dynamic DNS was standardized by RFC 2136, published in 1997, and was commonly used in conjunction with NAT to maintain address stability for dynamically addressed hosts. The emergence of mobile IP protocols, including those outlined in RFC 3344, further addressed the mobility issue by allowing devices to maintain the same IP address while moving between different networks.

d. Peer-to-Peer (P2P) Networking and NAT Traversal

In parallel, peer-to-peer (P2P) networking was becoming increasingly prevalent, with applications such as file sharing (e.g., Gnutella, Napster) and distributed computing. These systems often faced challenges with NAT traversal, as direct device-to-device connections were impeded by NAT devices. To overcome this, P2P applications used techniques like hole punching, where both peers behind NAT would simultaneously open a connection to a third-party server to facilitate direct communication.

These traversal techniques, while not perfect, were necessary for maintaining direct connectivity in decentralized networks. The ability to establish direct communication between peers despite NAT interference was a key feature for P2P systems, and methods to achieve this were well known and routinely employed.

e. Virtual LANs (VLANs) and Virtual IP Addresses

Another well-established concept was the use of Virtual LANs (VLANs) to logically segment a physical network into smaller, manageable sub-networks. VLANs allowed devices to communicate as if they were on the same local network, even if physically separated. This logical grouping of devices was fundamental for both security and network management.

Similarly, the concept of virtual IP addresses was commonly used to represent services or devices within a network. Virtual IPs were not tied to a single physical interface, allowing systems to maintain service availability even if the underlying hardware changed. Load balancers and failover systems routinely used virtual IPs to distribute traffic or provide redundancy.

f. Conclusion

By March 2003, the landscape of network communication and secure virtual networking was marked by mature, well-understood technologies. VPNs provided secure data transmission over public networks, NAT allowed efficient IP address management and connectivity, and dynamic DNS addressed mobility and dynamic IP assignment. P2P networking dealt with NAT traversal challenges using established methods, while VLANs and virtual IPs were commonplace for managing complex network topologies.

The '785 Patent, therefore, builds on these conventional techniques without introducing any novel technological solution. Its reliance on well-known methods for secure communication, NAT traversal, and virtual addressing does not advance the state of the art. Rather, it merely applies conventional networking practices to the problem of secure virtual community formation, which had already been addressed through routine methods available at the time.

2. The Asserted Claims of the '785 Patent Recite an Abstract Idea

The Asserted Claims of the '785 Patent are directed to the abstract idea of membership management and address look-up in a communications network—mere administrative tasks performed every day by receptionists, security personnel, and bouncers. Claim 30, the only independent Asserted Claim of the '785 Patent, recites:

[30.0] A virtual network manager, comprising:

[30.1] a network interface configured for data communication via a virtual network that is defined by a domain name having an associated public network address;

[30.2] a memory and a processor to implement a register module configured to register devices in a virtual network, the register module further configured to:

[30.3] receive a registration request from an agent associated with a device;

[30.4] distribute a virtual network address to the device when the device is registered in the virtual network, the device being identified to other devices in the virtual network by the virtual network address; and

[30.5] a DNS server for the virtual network, the DNS server configured to receive a DNS request from a first device in the virtual network, and return a network address associated with a network route director, a private network address associated with a second device in the virtual network, and a virtual network address associated with the second device.

Stripped of jargon, the claimed elements are basic administrative functions involved in setting up and maintaining a private communication network—operations that map directly onto longstanding abstract concepts such as maintaining a directory of users, assigning identifiers, and responding to queries with directing information. Claim 35 builds on claim 30 by adding a "join module" that receives join and leave requests—analogous to a sign-in/sign-out log for network members. Claim 37 further adds that the join module maintains a mapping between virtual addresses and devices—a conventional association table akin to a user directory or ARP (address resolution protocol) table.

The Asserted Claims of the '785 Patent all reflect the abstract idea of membership management and address look-up in a communications network—akin to a corporate office

receptionist managing an internal directory. The receptionist receives visitors (devices) requesting access (registration), checks whether they're authorized to enter (authentication), assigns them a visitor badge with an internal extension number (virtual address), and, when asked, tells other employees where to find the visitor (DNS resolution and routing). The receptionist also tracks when the visitor checks in and out (join and leave requests) and keeps a temporary log mapping badge numbers to visitors (address-device association). These are routine administrative functions that merely facilitate communication within a defined space—just as the Asserted Claims do within a virtual network. There is no innovation in how the receptionist performs these tasks, and these tasks have been performed by humans for centuries. Likewise, the Asserted Claims do not improve any underlying networking technology or propose any inventive technical implementation. The Asserted Claims of the '785 Patent are simply automating a familiar organizational role using conventional computing tools, rendering them abstract under *Alice* step one.

Tellingly, the method of originally filed claim 64 (which issued as claim 48) did not include any requirement that the steps of the claim be performed by a computer. On February 4, 2009 (5 years before the *Alice* decision), the Examiner rejected claim 64 under 35 U.S.C. § 101, applying the *Bilski* "machine-or-transformation" test for patentable subject matter that existed in 2009. In other words, without the recitation of a "machine" (*i.e.*, a computer), the claims were not patentable under § 101. In response to the Section 101 rejection, the Applicants amended claim 64 to be "computer-implemented," as none of the originally-filed claims required the use of a computer. Although an amendment requiring that the method be performed by a computer was sufficient in 2009-2010 to overcome the Examiner's Section 101 rejections, this is not the case today. Under current Supreme Court and Federal Circuit law, the *Alice* steps apply.

The '785 Patent Applicants' computer-implemented amendment to independent claim 64 crystallizes two points. First, the amendment demonstrates that the Applicants understood that the method described in the '785 Patent and claimed in the originally-filed claim 64 could be practiced without the use of a computer, *i.e.*, the method could be performed in the human mind, or by a human using a pen and paper. *See CyberSource Corp. v. Retail Decisions, Inc.*, 654 F.3d 1366, 1372–73 (Fed. Cir. 2011) (holding that a claim whose "steps can be performed in the human mind, or by a human using a pen and paper" is directed to an "unpatentable mental process[]"); *Elec. Power Grp.*, 830 F.3d at 1355 ("[C]laims focusing on organizing, characterizing, and/or sorting information are patent-ineligible because they are not significantly differentiable from ordinary mental processes."). Second, this amendment demonstrates that the focus of the claims of the '785 Patent is not on any specific asserted improvement in computer capabilities. *In re TLI Commc'ns L.L.C. Pat. Litig.*, 823 F.3d at 612 ("[T]he claims were not directed to specific improvement to computer functionality. Rather, they are directed to the use of conventional or generic technology in a nascent but well-known environment, without any claim that the invention reflects an inventive solution to any problem presented by combining the two.").

Under *Alice*, a claim that is unpatentable without the recitation of a computer does not become patentable by reciting a computer. That the claims of the '785 Patent merely implement the non-statutory subject matter on a generic computing device is plain from the fact that the originally-filed claims did not require that the claimed systems and methods include or be performed by a computer, respectively. Thus, the language of the Asserted Claims, their prosecution history, and the change in the law demonstrate that the Asserted Claims of the '785 Patent are directed to an abstract administrative idea applied to a communications network.

3. The Asserted ‘785 Patent Claims Lack an Inventive Concept

The Asserted Claims of the ‘785 Patent fail at *Alice* step 2 because they do not recite anything beyond generic computer functions performed using conventional components. The specification confirms that the system is built using well-known networking elements (*e.g.*, processors, memory, interfaces, DNS servers, NAT traversal techniques, virtual IP addressing) without any specialized circuitry, protocol, or configuration.

The ‘785 Patent itself acknowledges that the components and methods it employs were conventional at the time of invention. For example, the specification discusses the use of Network Address Translation (NAT) and virtual addresses to facilitate secure communication, but these techniques were already well-known in the field of networking. The patent does not propose any novel way of utilizing NAT or managing virtual addresses beyond their ordinary and conventional uses.

Claim 30, for instance, merely describes using a DNS server to resolve device addresses within a virtual network, which is a standard function of DNS servers generally. The idea of assigning virtual addresses to devices and resolving these addresses through DNS lookups is not unique or inventive. Instead, the patent recites high-level results—such as secure communication—without specifying how these results are technologically achieved. There is no disclosed protocol, no new addressing scheme, no algorithm for conflict resolution, and no structural innovation in how devices are registered or managed. The system instead relies on existing IP, UDP, NAT, and IPsec technologies. The claims are results-oriented: “register the device,” “assign an address,” “respond to a DNS query”—without any accompanying inventive technical solution.

The claim elements—processor, memory, network interface, modules—are standard hardware or logical software components performing their expected functions. There is no claim

that the invention requires new hardware, reconfigured components, or even custom protocols. The Asserted Claims of the '785 Patent are therefore equivalent to the claims in *BSG Tech LLC v. BuySeasons, Inc.*, 899 F.3d 1281, 1291 (Fed. Cir. 2018), where the claims at issue merely limited an abstract idea to a particular technological environment and were held ineligible. Similarly, in *ChargePoint, Inc. v. SemaConnect, Inc.*, the court invalidated claims that combined generic networking capabilities with existing concepts without adding any inventive technology. 920 F.3d 759, 774 (Fed. Cir. 2019) ("But network control is the abstract idea itself, and 'a claimed invention's use of the ineligible concept to which it is directed cannot supply the inventive concept that renders the invention 'significantly more' than that ineligible concept.'") (quoting *BSG Tech*, 899 F.3d at 1290). The Asserted Claims of the '785 Patent follow the same pattern by applying well-understood networking techniques without advancing the state of the art.

4. Conclusion

The Asserted Claims of the '785 Patent are invalid under 35 U.S.C. § 101. They are directed to abstract administrative operations for managing virtual networks and fail to recite any inventive concept in either their individual components or overall arrangement. They do not improve any underlying technology, nor do they disclose a novel technical mechanism for achieving the stated goals. As such, they fail both steps of the *Alice* test and are ineligible for patent protection.

5. Admissions in the Specification

a. Network Address Translation (NAT) as Routine and Conventional:

The specification of the '785 Patent explicitly acknowledges that NAT (Network Address Translation) was a well-known and conventional technique at the time: "One popular solution to the depleting address problem is Network Address Translation (NAT). This concept includes predefining a number of network addresses to be private addresses and public addresses. Public addresses are unique addresses that should only be used by one entity having access to the Internet.

That is, no two entities on the Internet should have the same public address. Private addresses are not unique and are typically used for entities not having direct access to the Internet. Private addresses are used in private networks. Private addresses can be used by more than one organization or network. NAT assumes that all of the machines on a network will not need to access the Internet at all times. Therefore, there is no need for each machine to have a public address." 785 Patent at 1:56-62.

b. Virtual Private Network (VPN) as Well-Known:

The patent also identifies VPN technology as an existing and established solution: "One solution currently in use that provides for local and remote specified entities to communicate and collaborate using the Internet is the Virtual Private Network (VPN). The VPN uses additional network software layers to increase security between users in the public realm and those in private realms. For example, some VPNs encrypt packets using IPsec (or other protocols). The encrypted packets are then encapsulated within a standard packet and transmitted across the Internet to the destination. At the destination, the encrypted packet is decrypted." 785 Patent at 4:58-67.

c. Dynamic DNS as Known and Conventional:

Dynamic DNS, which is also mentioned in the patent, is described as an existing technology that predates the invention: "One method of overcoming the mobility problem includes the use of Dynamic DNS. More information about Dynamic DNS can be found in RFC2136 incorporated herein by reference." 785 Patent at 5:11-14.

d. Hardware and System Components as Conventional:

The specification explicitly states that the hardware components required for the invention were standard and understood: "A hardware architecture for the machines, server or other devices used to implement the present invention should be well understood to one of average skill in the art. Suitable hardware includes one or more processors, a memory, a mass storage device, a

portable storage device, a first network interface, a second network interface and I/O devices, in communication with each other." 785 Patent at 9:42-48.

e. IP Addressing and Network Protocols as Routine:

The specification discusses the standard nature of IP addressing and network protocols: "Most machines on the Internet use the TCP/IP (Transmission Control Protocol/Internet Protocol) reference model to send data to other machines on the Internet." 785 Patent at 3:1-3.

D. U.S. Patent No. 8,027,326

Representative Claim 1 recites:

> [1.0] A method for increasing data rates and data throughput in a network, the method comprising:
>
> [1.1] selecting at least a first channel and a second channel, wherein the first channel and the second channel are adjacent without any other channels there between, wherein the first channel and the second channel each have a plurality of data subcarriers, wherein the data subcarriers of the first channel and the data subcarriers of the second channel are separated by a frequency gap corresponding to one or more guard bands between the first and second channels;
>
> [1.2] partially filling the frequency gap between the first channel and the second channel by adding one or more data subcarriers into the frequency gap such that the one or more guard bands are at least partially filled with at least some of the one or more data subcarriers using full spectral synthesis capability of a fast fourier transform or an inverse fast fourier transform;
>
> [1.3] **combining the first channel and the second channel** using channel bonding with orthogonal frequency division multiplexing (OFDM); and
>
> [1.4] transmitting data subcarriers occupying the first channel, the second channel, and the frequency gap in parallel to a receiver.

1. State of the Art Relevant to the '326 Patent

Southwest sets forth a summary of its current understanding of the state of the art for the general subject matter of the '326 Patent. The '326 Patent generally relates to a "method and

system for high data rate multi-channel wireless communications" which reuses "existing single channel radio designs" to "achieve very high data rate expanded bandwidth (wide band) WLAN operations." Broadly, the '326 Patent concerns the development of more efficient Wi-Fi, with I.V.'s Complaint focusing on "hardware and/or software from these providers that support Wi-Fi, including IEEE 802.11n and 802.11ac protocols." Dkt#1 ¶ 93. These Wi-Fi protocols were widely used as of the date of filing:

"802.11ac was finalised in 2013 and you will find it in every major smartphone, laptop and desktop computer and smart television. It succeeds the equally-badly named '802.11n' which has been around since 2007 and brings some major benefits." Kelly, Gordon, "802.11ac vs 802.11n Wi-Fi: What's the Difference?" Forbes, December 30, 2014 *available at* https://www.forbes.com/sites/gordonkelly/2014/12/30/802-11ac-vs-802-11n-wifi-whats-the-difference/.

802.11 was already a popular standard. "IEEE 802.11-based wireless technology is well known by the common name 'Wi-Fi' (Wireless Fidelity) and provides wireless connectivity to the majority of today's digital devices such as mobile phones, laptops, smart TVs, etc. The most commonly used wireless technologies in smart manufacturing, especially discrete manufacturing industries are also IEEE 802.11 based." Liu, Xuejing, et al., "Exploring the Regulatory Mechanisms of Long Non-Coding RNAs in the Pathogenesis of Atherosclerosis," Pubmed Central, August 1, 2022 *available at* https://pmc.ncbi.nlm.nih.gov/articles/PMC9824593/.

Trends in the Wi-Fi industry included improving internet speeds, increasing bandwidth and reducing interference, particularly as the volume of users increased.

2. The Asserted Claims of the '326 Patent are Ineligible Under 35 U.S.C. § 101

The Asserted Claims (claims 1, 4, and 18) of the '326 Patent are directed to the abstract idea of combining adjacent communication channels to increase data throughput in a network. The Asserted Claims of the '326 Patent merely implement this concept using conventional networking techniques and generic computing components, offering no technological innovation or improvement. As such, they are ineligible under 35 U.S.C. § 101 because they are directed to an abstract idea and fail to recite an inventive concept, as required under the two-step framework set forth in *Alice Corp. v. CLS Bank Int'l*, 573 U.S. 208 (2014).

a. The Asserted Claims of the '326 Patent are Directed to an Abstract Idea

The Asserted Claims of the '326 Patent recite the abstract idea of combining channels to increase data throughput (claims 1 and 18) and transmitting data to multiple radios (claim 4), and fail Step 1 of *Alice*. *See Alice*, 573 U.S. at 216-17. These are long-standing and fundamental practices in sending and receiving information over a network—not a technological improvement or specific technical solution. There is nothing to the patent beyond the concept of 'combine two pipes together, including the space between them, to get a bigger pipe.'

Channel bonding and transmitting data to multiple radios is an abstract idea as claims directed to sending and receiving data are abstract in nature. *See, e.g., Affinity Labs of Texas, LLC. v. DIRECTTV, LLC*, 838 F.3d 1253, 1261 (Fed. Cir. 2016) (explaining that patents directed to "the conveyance and manipulation of information using wireless communication and computer technology" do not disclose eligible subject matter). Claims directed to formatting data are also abstract. *See Dropbox, Inc. v. Synchronoss Techs., Inc.*, 815 F. App'x 529, 537 (Fed. Cir. 2020) (finding claims ineligible that were directed, in part, to "formatting data").

b. The ‘326 Patent Claims Lack an Inventive Concept

Step 2 of *Alice* cannot save the Asserted Claims of the ‘326 Patent from ineligibility under 35 U.S.C. § 101. The Asserted Claims of the ‘326 Patent do no more than implement the abstract idea of increasing data throughput by using generic computing components performing routine, conventional functions. Namely, channel bonding (claims 1 and 18) and transmitting data to multiple radios (claim 4) lack an inventive concept because such claims amount only to generic components and fail to demonstrate unconventionality. *See Dropbox*, 815 F. App’x at 537 (finding that “data transfer connection[s] are generic computer-related concepts as they literally are just data transfer modalities” and are thus not inventive concepts).

c. Conclusion

The Asserted Claims of the ‘326 Patent are invalid under 35 U.S.C. § 101. They are directed to the abstract idea of transferring data over a network and fail to recite an inventive concept. Because the claims implement an abstract idea using only generic networking and data-handling functions without reciting any technological improvement or unconventional mechanisms, the claims fail both steps of the *Alice* framework.

E. U.S. Patent No. 7,324,469

The ‘469 Patent, titled “Satellite Distributed High Speed Internet Access,” claims methods for providing an internet hotspot to subscribers in a remote location. Asserted independent Claim 24 of the ‘469 Patent recites:

> 24. An Internet Hotspot comprising:
>
> a satellite dish **communicating with the Internet** via one or more data links with a satellite;
>
> at least one router operatively coupled to the satellite dish;

> a subscriber access unit operatively coupled between the satellite dish and the at least one router, the subscriber access unit **being capable of authenticating** a subscription account associated with a user prior to allowing the user access to the Internet; and
>
> a web-ready device operatively coupled to the at least one router, the web-read device **having a browser application** operating thereon for accessing the Internet;
>
> wherein the satellite dish, at least one router and the subscriber access unit are located in a remote location a experiencing a relatively high volume of transient traffic;
>
> wherein the user **may authenticate** the subscription account and access the Internet at the remote location by **establishing a data connection** between the web-ready device and the router.

(emphasis added).

1. State of the Art Relevant to the '469 Patent

As of September 2003—the earliest claimed priority date of the '469 Patent—wireless and wired Internet hotspots were ubiquitous, and their operation as a pay-for-use business was well known. For example, a 2003 book entitled "Going Wi-Fi" describes that hotpots were "all the rage":

> North America, Singapore, Korea and Japan have led this Wi-Fi trend, offering traveling laptop users temporary wireless Internet access in various types of public gathering spots including, for example, hotels and cafés. According to Garner Inc., the number of HotSpots will grow to more than 71,000 in 2003 from about 1200 in 2001.

Reynolds, Janice, "Going Wi-Fi A Practical Guide to Planning and Building an 802.11 Network," CMP Books, San Francisco (2003) ("Going Wi-Fi") at 4. These thousands of Internet hotspots included monthly subscription-based service as well as some that "allow users to purchase as little as 30 minutes worth of wireless access for the equivalent of a few dollars." *Id*. at 5. Well-established telecommunications providers like T-Mobile and Boingo were, at the time the Asserted Patent was filed, already offering hotspots "from Starbucks to hotels to airports." *Id.* ("and, of course, they plan to charge plenty for it."). Numerous publications describe the availability of hotspots at locations such as hotels, airports, coffee shops, and fast-food restaurants. *See* Ellison,

Carol, "Unwire Everywhere," PC MAGAZINE (Special Wireless Issue 2003) at 97-104 ("Hot spots are cropping up at a remarkable rate to meet the connectivity demands of users as they venture out from their homes and offices.").

The concept of subscription or payment authorization to access Internet hotspots was likewise well established at the time of the filing of the '469 Patent, and many companies offered "turnkey" hotspot "kits" which

> deliver the fundamental services layer by providing standards-based services such as secure authentication, automated configuration, IP routing, firewalls, caching, remote management, usage tracking, and bandwidth shaping.

Going Wi-Fi at 271.

With the advent of satellite Internet connectivity quickly came satellite Internet "hotspots,"—of course, for a fee. In one example, Airpath Wireless offered turnkey hotspot hardware, software, and connectivity services "for Hot Spot providers in need of broadband connectivity in remote or cost-prohibitive Hot Spot locations," such as "marinas, truck-stops, campgrounds, resorts, and hotels located in rural areas." June 25, 2003, Press Release available at https://web.archive.org/web/20030808000849/http://www.mainstreamdata.com:80/content/_news/news0.html. Airpath services included a "Hot Spot Billing Suite" for authentication and subscriber management. Satellite networks were especially suited to implementation of hotspots outside of urban settings, including military bases throughout the world. iDirect Government provided Internet cafes in US military tents and buildings around the world beginning in March 2000. *See* https://www.idirectgov.com/media/1jxdg0c3/idigt_cs_spawar-mwrnet_0215.pdf.

Although the '469 Patent does not mention aircraft or in-transit connectivity—the patent addresses only fixed locations such as "conference centers"—Southwest also addresses the state of the art pertaining to in-flight connectivity in view of the plaintiff's construction of the claims to

encompass in-flight connectivity systems. The Boeing Company announced its in-flight connectivity system, "Connexion," three years before the filing of the provisional application of the '469 patent, in April 2000. Building on its proprietary satellite transceiver antenna technology, Connexion installed a "flat, two-inch antenna and additional wiring [] to each top of aircraft" which "provides enhanced response to directional changes" and which adjusted "to keep in touch with the nearest satellite." Suwanateep at 7-8. Boeing's Connexion system was among the first to offer real-time airborne connectivity. *Id*. at 5.

> The cabin is equipped with onboard server, router and service-analysis equipment and is outfitted for the security enhancement capabilities pursuing for commercial airline customers. Seats are configured for connectivity. Since Internet traffic tends to happen in bursts, the cabin file-server not only allows multiple users to share access to a single physical connection, but also allows them to be on-line simultaneously.

Suwanateep at 9. Connexion by Boeing was first installed on private corporate aircraft, but later rolled out on major commercial airlines including United Airlines and Delta in early 2003. With commercial airline adoption came paid access for travelers using Connexion In-Flight Portal:

Find out about our In-Flight Portal



Before you access the Internet through the Connexion by Boeing service, you will have the option to select a variety of free services and content through our In-Flight Portal.

The In-Flight Portal's selection of top news and entertainment headlines will help keep you up to date, even at 39,000 feet. You can also use the In-Flight Portal to register for the service, view Frequently Asked Questions (FAQs), and contact customer care.

https://web.archive.org/web/20030201115424/http://www.connexionbyboeing.com/index.cfm.

Aircrafts were not the only in-transit systems for Internet access designed to meet growing demand in the early 2000s. In 2002, satellite provider Intelsat partnered with Maritime Telecommunications Network (MTN) to provide connectivity to maritime customers including Norwegian Cruise Lines. *At Sea But Never Out Of Touch*, CRUISE INDUSTRY QUARTERLY,

(Summer 2003) at 25. MTN provided complete hardware and software solutions to cruise ships, including internet cafes and billing options for passenger access.

2. The Asserted Claims of the '469 Patent recite the business practice of placing a paid hotspot in a "transient location"

Each of the Asserted Claims of the '469 Patent are invalid under 35 U.S.C. § 101 because the claims are directed to an abstract idea and fail to describe an inventive concept. *Alice Corp. v. CLS Bank Int'l*, 573 U.S. 208, 216-17 (2014). For example, all of the Asserted Claims of the '469 patent are generally directed to an abstract idea for Section 101 purposes and under Intellectual Venture's apparent constructions. The sole asserted independent claim, claim 24, concerns placing a known composition of commercial components in a "remote location" to establish a subscribers-only internet hotspot. The specification also makes clear that the invention of the Asserted Patent is the idea of placing paid hotspots where users would otherwise not have Internet access:

> The present invention provides rural "Hotspots" (such as Wi-Fi access, for example) to enable wireless and hardwired, satellite distributed Internet access for anyone with a PC or other web-ready device (wireless ready or cabled) and a valid credit card.

'469 Patent at [1:34-38]. The business idea of charging people money for internet access in a "transient location" like a restaurant or motel is just that: a business idea. Paying for services is a "fundamental economic practice long prevalent in our system of commerce" and, "as such, is a patent ineligible abstract idea." *W. Express Bancshares, LLC v. Green Dot Corp*., 816 F. App'x 485, 487 (Fed. Cir. 2020).

Applying this fundamental economic practice to tangible components such as commercially available network equipment does not alter the fact that the claims are directed to "an art or principle in the abstract, and not for any particular method or machinery." *ChargePoint, Inc. v. SemaConnect, Inc.*, 920 F.3d 759, 769 (Fed. Cir. 2019) (quoting *Wyeth v. Stone*, 30 F. Cas. 723 (C.C.D. Mass. 1840)); *ChargePoint*, 920 F.3d at 770 ("In short, the inventors here had the

good idea to add networking capabilities to existing charging stations to facilitate various business interactions. But that is where they stopped, and that is all they patented. We therefore hold that claim 1 is 'directed to' an abstract idea"). The patentee's implementation rests wholly on the existing knowledge of a person of skill in the art: "The Hotspot 10 comprises *all the equipment installed at the rural location that is necessary to provide Internet access via satellite*." '469 patent at 4:18-21 (emphasis added). "The Ethernet wireless access point 40 can employ *a security protocol of a type that is known to persons skilled in the art*, such as the 802.11b security protocol, or any subsequent or future versions of the 802.11 standard." '469 patent at 5:19-25 (emphasis added). The Federal Circuit has "repeatedly found the concept of controlling access to resources via software to be an abstract idea." *Ericsson Inc. v. TCL Commc'n Tech. Holdings Ltd*., 955 F.3d 1317, 1327 (Fed. Cir. 2020); *see also Smartflash LLC v. Apple Inc.*, 680 F. App'x 977 (Fed. Cir. 2017).

3. The Asserted Claims of the '469 Patent lack an inventive concept.

The specification provides no solution to overcome technical challenges associated with "remote"-ness, or a "high volume of transient traffic" or "subscriber access." The claims of the Asserted Patent are results-driven and fail to overcome ineligibility. Each limitation recites generic components which are "capable of" performing connectivity, routing, and authorization using any method whatsoever. As set forth above and in the attached claim charts, paid Internet hotspots were very well known and widely used at the time of the Asserted Patent. Placing a paid Internet hotspot in "a remote location" is insufficient to render the claims eligible where no particular improvement is recited to, for example, teach *how* a subscriber access unit is "capable of" authorizing payments in "a remote location." *Universal Secure Registry LLC v. Apple Inc*., 10 F.4th 1342, 1346 (Fed. Cir. 2021); *Elec. Power Grp., LLC v. Alstom S.A.*, 830 F.3d 1350, 1354 (Fed. Cir. 2016) ("limiting the claims to the particular technological environment of power-grid

monitoring is, without more, insufficient to transform them into patent-eligible applications of the abstract idea at their core."). The "essentially result-focused, functional character of claim language has been a frequent feature of claims held ineligible under § 101, especially in the area of using generic computer and network technology to carry out economic transactions."

Each of the asserted dependent claims likewise recite the use of conventional components or processes to achieve results for which those components are designed. '469 Patent, claim 25 ("a wired data connection" "a wireless data connection"); claim 26 ("a plurality of users **may access** the Internet simultaneously," "at the remote location," "establishing data connections with the router via their web-ready devices"); claim 28 ("the data connections include wireless data connections"); claim 32 ("the wireless connection," "802.11a wireless area network," "802.11b wireless area network," "802.11g wireless area network," "802.11n wireless area network"). *See also CyberSource Corp. v. Retail Decisions, Inc*., 654 F.3d 1366, 1370-73 (Fed. Cir. 2011) (a method of verifying the validity of credit card transactions over the Internet held patent-ineligible as directed to an abstract idea); *buySAFE, Inc. v. Google, Inc*., 765 F.3d 1350, 1354-55 (Fed. Cir. 2014) (guaranteeing performance in an Internet transaction was patent-ineligible); *TLI Communications LLC Patent Litigation*, 823 F.3d 607, 615 (Fed. Cir. 2016) ("vague, functional descriptions of server components are insufficient to transform the abstract idea into a patent-eligible invention."). The claimed steps are described at a high level of generality, using generic terms, confirming the '469 is ineligible under 35 U.S.C. §101.

F. U.S. Patent No. 7,257,582

1. State of the Art Relevant to the '582 Patent in March 2002

The principles underlying the '582 Patent's claims were widely known and implemented in numerous systems prior to the claimed priority date of March 13, 2002. Long before March 2002, distributed computing frameworks had already embraced first-come, first-served dynamic

task assignment, self-scheduling workloads, and efficient data partitioning across multiple processing nodes. These methods were integral to parallel computing, volunteer computing initiatives, and large-scale database management—fields in which researchers had been refining these concepts for over a decade prior to March 2002.

By the late 1990s, high-performance computing ("HPC") systems, parallel databases, and distributed file systems were actively employing automated load balancing techniques, allowing processors to dynamically claim tasks as they became available. The master-worker paradigm, in which a master process assigns work to multiple processors and redistributes tasks based on availability, was a foundational computing concept. Volunteer computing projects like SETI@home (1999) had already demonstrated the real-world application of parallel task distribution and result aggregation, processing enormous datasets by dynamically distributing work units to thousands of participating machines. Similarly, industry-standard MPP (massively parallel processing) databases had been using data partitioning techniques since the 1980s to distribute processing workloads across multiple nodes, enabling efficient parallel query execution.

These well-established techniques, which had been publicly disclosed and successfully deployed in both academic and industry settings, directly contradict any claim that the '582 Patent introduced a novel approach. The state of the art at the time confirms that the patent merely recites preexisting, widely implemented concepts in distributed computing and parallel processing.

a. Dynamic Scheduling in Parallel Processing was Well Known by March 2002.

By the early 1990s, dynamic scheduling of computational workloads was an established technique in parallel computing. Long before March 2002, researchers and industry practitioners had widely implemented first-come, first-served task scheduling to efficiently utilize processor resources. The master-worker paradigm—where a central process partitions a workload and

distributes it dynamically to available processors—was already a fundamental principle of parallel computing systems. Further, the bag-of-tasks model, introduced by Carriero and Gelernter in the late 1980s, demonstrated how worker processes could independently pull new tasks from a shared task pool, ensuring that faster processors were never idle while waiting for slower ones. This dynamic, demand-based task assignment allowed distributed computing systems to scale efficiently and optimize processor utilization.

By 1990, researchers had expanded on this concept through self-scheduling for parallel loops, where processors independently claimed computational iterations rather than following a static, predetermined allocation. Hewlett-Packard's 1994 patent, U.S. Patent No. 5,325,493, further reinforced this technique by describing a system in which processors would automatically request and execute tasks on a first-come, first-served basis, eliminating bottlenecks and ensuring that computational work was dynamically distributed. In that system, each processor, upon finishing its current task, automatically requests the next task, and processed data is eventually recombined in order. The patent emphasizes that this approach maximizes parallel throughput by not waiting for slower processors.

The same principle was later embodied in volunteer computing projects such as SETI@home (1999), which partitioned large astronomical data files into discrete work units and assigned them to available processors across a network. Each participant's system would request the next available task, execute it, and return results asynchronously to be merged into the central database. This demonstrated not only the viability but also the widespread public deployment of dynamically distributed task execution in a distributed computing environment. By 2000, Intel's patent application (US 2002/0040381 A1, filed October 2000) described a multi-processor load-balancing system that placed computational tasks into a queue, where processors would pull the

next available job upon completion of a prior task, ensuring continuous processor utilization. This system functioned precisely as described in the ‘582 Patent: tasks were not assigned in advance but dynamically distributed as processors became available.

These techniques were not confined to academic research—they were actively deployed in high-performance computing, commercial database processing, and networked data analysis. Systems such as IBM LoadLeveler, Sun Grid Engine, and Condor were already widely used in enterprise computing to dynamically distribute jobs among processors. These systems employed task scheduling policies, including first-come, first-served scheduling, priority-based scheduling, and adaptive load balancing, ensuring optimal utilization of processing resources. The ‘582 Patent describes an approach where subtasks are executed in parallel and reassigned based on availability, but this was precisely the methodology employed by Condor—a job scheduling system developed at the University of Wisconsin, which allowed idle processors to claim unassigned tasks dynamically. Similarly, the Globus Toolkit, an essential framework for grid computing, provided mechanisms for distributed task execution across heterogeneous systems, allowing real-time reassignment of workloads based on processor capacity. Google’s 2004 MapReduce paper would later credit these well-established techniques as the foundation for large-scale distributed computing, confirming that the core scheduling method claimed in the ‘582 Patent was well-known and widely implemented before March 2002

The ‘582 Patent itself acknowledges that job scheduling in parallel systems was well understood, admitting that dynamic task execution techniques were already known in the art. The first-come, first served scheduling policy, in particular, was not a novel concept in the context of multi-processor computing; rather, it was a basic scheduling principle applied in numerous high-performance computing environments. U.S. Patent No. 6,334,137 (2001) further reinforces this

point, as it describes a system for controlling the parallel execution of jobs, dynamically registering jobs, and balancing workloads among multiple computing nodes. Similarly, U.S. Patent No. 5,765,166 (1998) highlights a parallel processing approach for multiple hypothesis tracking, which includes task distribution based on processor availability, mirroring the claim structure of the '582 Patent. Thus, dynamic scheduling as described in the '582 Patent was already well established in both commercial products and prior patents, confirming that the claimed invention lacks novelty.

b. Data Partitioning and Shared Processing was Well Known by March 2002.

By March 2002, parallel data processing had been an essential component of high-performance computing for decades. In the 1980s, massively parallel processing (MPP) databases such as Teradata had introduced data partitioning, where large datasets were automatically divided across multiple processors for simultaneous execution. Research projects such as the Gamma database further developed this concept, refining techniques for parallel query execution on partitioned data. These advances laid the foundation for modern distributed data processing systems and demonstrated that splitting an input dataset into independently processed partitions was a well-established practice.

The concept of shared file access across multiple processors was equally well understood. The Andrew File System (AFS), developed in the 1980s, allowed multiple machines to access and process the same file concurrently, enabling parallel computation on shared datasets. By the late 1990s, enterprise computing environments had fully embraced Network Attached Storage (NAS) and Storage Area Networks (SAN), providing high-speed, concurrent access to large datasets by multiple processing nodes. These technologies made it trivial for a cluster of machines to read from and write to a shared file, a key capability claimed in the '582 Patent.

Meanwhile, volunteer computing projects such as SETI@home and distributed.net (1997) were actively employing data partitioning and parallelized computation across thousands of machines worldwide. SETI@home, for example, split radio telescope data into work units, distributed them to participating computers, and recombined results asynchronously, demonstrating large-scale parallel processing on shared datasets. Similarly, distributed.net divided large cryptographic keyspaces into smaller search ranges, assigned them to individual processors, and collected results into a master database—a direct precursor to the distributed workloads envisioned in the '582 Patent.

The process of dividing large computational tasks across multiple processors and merging the results was also well established in high-performance computing (HPC). Systems such as IBM's LoadLeveler and Condor (1990) scheduled jobs across clusters of computers, often breaking down large computational tasks into parallelized workloads. Parallel database management systems, including those developed by Oracle and IBM, routinely split and processed massive tables across multiple processors to accelerate query performance. These well-documented techniques anticipated the purported invention of automatically partitioning an input dataset, distributing partitions across processors, and merging the results into an output dataset.

Further, prior to March 2002, Message Passing Interface (MPI) and Parallel Virtual Machine (PVM) had become the standard for distributed computing, providing detailed methodologies for logically subdividing workloads across multiple nodes. These frameworks enabled data segmentation, real-time allocation of tasks to available processors, and efficient reassembly of processed outputs. Similarly, Parallel Database Management Systems (DBMS), including Oracle Parallel Server and IBM DB2, had implemented partitioning strategies for decades, allowing concurrent query execution across multiple computing nodes. Likewise, U.S.

Patent No. 5,535,393 (1996) discloses a system for optimally distributing workloads among processors while minimizing data contention—fundamentally the same goal the '582 Patent seeks to achieve. U.S. Patent No. 6,304,866 (2001) discusses a multiprocessing system employing on-demand and incremental sub-task allocation, further demonstrating that the concept of dynamically segmenting data was already well known. The notion that file processing can be subdivided logically and executed in a distributed manner was, therefore, a standard principle in computing prior to March 2002.

c. Conclusion

Thus, the '582 Patent fails to introduce any novel aspect in how partitions are defined, stored, or processed. It relies on general functional language, describing the need to logically subdivide records without detailing any new approach or specific improvement over conventional systems. As of the March 2002 earliest possible priority date, the practice of splitting files for parallel execution—whether by byte ranges, record boundaries, or database partitioning schemes—was fully established in the art.

2. The '582 Patent Claims Ineligible Subject Matter Under 35 U.S.C. § 101

The Asserted Claims (claims 1, 2, 5, and 7) of the '582 Patent are invalid under 35 U.S.C. § 101 because they are directed to an abstract idea—specifically, the idea of dividing a large task into subtasks and processing them using available resources on a first-come/first-served basis—and lack any inventive concept that transforms this abstract idea into patent-eligible subject matter. The Asserted Claims of the '582 Patent therefore fail at both steps of the Alice test and are invalid. See *Alice Corp. v. CLS Bank Int'l*, 573 U.S. 208 (2014).

The ‘582 Patent, titled “Method for Managing Distributed Processing of Tasks in a Network,” claims a method for subtask processing in distributed computing systems. Claim 1—the only independent claim of the ‘582 Patent—recites:

> 1. A method of effecting on **a preexisting input file** a computer-executable process **comprised of a plurality of subtasks**, the method comprising the steps of:
>
> (a) automatically determining file allocation and **logically subdividing records of said input file into a plurality of partitions**;
>
> (b) **distributing descriptions of** all of said **partitions to** each of a plurality of **subtask processors**[;]
>
> (c) **simultaneously executing** at least a respective **one of the subtasks** of the computer-executable process in each of at least some of said processors on a respective one of the partitions with each subtask reading and processing the respective partition so as to process the respective partition and produce respective subtask output and;
>
> (d) thereafter **repeating step (c)** in at least some of the subtask processors each **with another unprocessed partition on a first-come/first-served basis**; and
>
> (e) generating at least one output **combining all of the subtask outputs** and reflecting the processing of all of said subtasks.

The patent admits that these steps mostly were well known in the art, including partitioning data, distributing tasks, parallel processing, and aggregating results. Dkt#1-6 at 3:35-47, 3:67-4:15, 4:59-63. Partitioning (step a) relies on conventional techniques, such as byte ranges or track addresses, while subtask distribution (step b) employs generic mechanisms using control files; processing subtasks (step c) involves standard read-process-write operations, and aggregation (step e) is “of course” a conventional merging operation. *Id.* at 3:35-4:15, 4:59-63. The dependent Asserted Claims of the ‘582 Patent fare no better. Claim 2 merely states that a separate processor may perform the partitioning and distribution—an obvious and conventional division of labor that lacks any inventive concept. Claim 5 recites that subtask outputs can be accumulated in arbitrary order but offers no technical detail or mechanism for doing so—just a result. Claim 7 limits the

input file to being stored on a network-attached storage device, which the patent itself identifies as a well-known technology.

The claimed first-come/first-served scheduling (step d) was argued during prosecution as the "primary difference" over the prior art. *See* Ex. A, Mar. 2, 2007 Remarks. According to the applicant, the prior art used "load information to distribute the load between processors," whereas:

> *With the instant invention … load information is not created* …. Instead, the load sharing is done as a byproduct of the fact that *the load-sharing process take parts of the load on a first-come/first-served basis*. A comparison would be to a road intersection where, according to the prior art, there is a traffic light that determines who can go when. The instant invention is more like such an intersection with a four-way stop so that the individual drivers determine who can go and when.

Id. The patentee's admission about the "primary difference" between the claimed invention and the prior art is critical in two ways. First, by pinpointing the "first available" assignment technique as the point of novelty, the inventors have defined what the claimed invention is "directed to" for purposes of the *Alice* Step 1 inquiry. *See Affinity Labs of Tex., LLC v. DIRECTV, LLC*, 838 F.3d 1253, 1257–58 (Fed. Cir. 2016) ("The 'abstract idea' step of the inquiry calls upon us to look at the 'focus of the claimed advance over the prior art' to determine if the claim's 'character as a whole' is directed to excluded subject matter."). Second, by the simple comparison between the invention and the managing of traffic loads at a street intersection, the inventor acknowledged that his "invention" merely applies a longstanding method of organizing human activities to computers.

The methods described in the '582 Patent implement that abstract concept using generic computing components, such as processors and memory, which are standard elements in distributed systems. The claims use simple functional language to describe desired outcomes—partitioning, distributing, processing, scheduling, and aggregating—without detailing how these

operations are performed or improved. Consequently, the Asserted Claims of the '582 Patent fail at both steps of the Alice test and are therefore invalid.

a. The Asserted Claims of the '582 Patent recite an abstract scheduling technique.

The Asserted Claims of the '582 Patent are directed to the abstract idea of dividing a large task into subtasks and processing them using available resources on a first-come/first-served basis. Claim 1 of the '582 Patent—the only independent claim of the '582 Patent—comprises five steps: (a) "logically subdividing records . . . into a plurality of partitions," (b) distributing descriptions of those partitions to multiple "subtask processors," (c) "simultaneously executing" at least one "subtask" on each of the subtask processors, (d) thereafter allocating any remaining "unprocessed partition" to the subtask processors "on a first-come/first-served basis," and (e) aggregating "all of the subtask outputs" into a final combined output. Stripped of jargon, Claim 1 covers the basic concept of splitting up a task and distributing the work, then collecting the results.

Claim 1's distributed task management approach is an abstract idea that humans have long practiced without computers; it is akin to breaking a large job into smaller tasks, handing those tasks out to a group of workers, giving each new work as they free up, and then assembling the final product. The practice of breaking up a large job into discrete tasks, passing out a first round of assignments to team members, instructing them to ask for a new assignment when finished, and then combining the results has been employed by humans since the building of the Egyptian pyramids. The Federal Circuit has consistently held that processes based on such fundamental organizational practices are abstract. *See, e.g.*, *Intellectual Ventures I LLC v. Capital One Bank (USA)*, 792 F.3d 1363, 1367 (Fed. Cir. 2015) (organizing, storing, and retrieving information held abstract); *Intellectual Ventures I LLC v. Symantec Corp.*, 838 F.3d 1307 (Fed. Cir. 2016)

(automatically organizing information into wanted and unwanted categories and discarding the unwanted information held abstract).

"[T]here is a critical difference between patenting a particular concrete solution to a problem and attempting to patent the abstract idea of a solution to the problem in general." *Electric Power Group, LLC v. Alstom S.A.*, 830 F.3d 1350, 1356 (Fed. Cir. 2016). The Asserted Claims of the '582 Patent fall on the latter side of that line: they seek to monopolize the general idea of a solution (parallel task processing with on-the-fly scheduling) rather than a concrete technological solution. The Asserted Claims of the '582 Patent do not improve computer technology or solve a specific technical problem in how computers operate but instead apply conventional and common-sense workload management ideas to distributed computing. Partitioning data, distributing tasks, processing in parallel, and aggregating results are standard operations in distributed computing systems. Dkt#1-6 at 3:35–4:15, 4:59–63. Simply executing a known business/workflow practice on generic computers is not a patentable improvement to those computers.

The '582 Patent's recitation of assigning work using first-come/first-served scheduling—a ubiquitous method of task processing akin to queue management (e.g., a line at a concession stand or a four-way stop)—demonstrates its abstract nature. Claim 1 applies this principle to parallel computer processing without offering any new technological innovation. Moreover, The Asserted Claims of the '582 Patent use abstract functional language to describe desired outcomes—such as "logically subdividing," "distributing," and "aggregating"—without detailing how these steps are performed. This kind of result-oriented claiming that specifies an outcome (efficient parallel processing) without limiting the claim to any particular innovative technique for achieving it is a hallmark of an abstract idea. Thus, the '582 Patent "merely applies a well-known idea using generic computers 'to the particular technological environment of the Internet.'"

Intellectual Ventures I LLC v. Symantec Corp., 838 F.3d 1307, 1314 (Fed. Cir. 2016). In *Two-Way Media*, the Federal Circuit held that reciting a series of steps to achieve a desired result, without specifying how those steps are performed, renders the claim abstract. *See Two-Way Media Ltd. v. Comcast Cable Commc'ns*, LLC, 874 F.3d 1329, 1337–38 (Fed. Cir. 2017) ("routing" and "accumulating records" held abstract and invalid). Even though the specification describes conventional methods for these operations, the Asserted Claims of the '582 Patent do not specify how data is partitioned, how partitioned subtasks are distributed, or how results are aggregated. Dkt#1-6 at 3:35–4:15, 4:59–63. Such high-level, results-oriented language underscores the claim's abstract nature. As the Federal Circuit noted in *Electric Power*, claims that focus on the outcome of data processing without specifying the means to achieve it are abstract and invalid. 830 F.3d at 1356.

IV argues that the Asserted Claims of the '582 Patent are not abstract because they are "directed to an improvement in data processing"—specifically claiming that using multiple computers in parallel "increases the speed of execution of processes and balances the load" on resources. Dkt#28 at 9-10. The claimed method only achieves these benefits by applying the abstract idea of task partitioning and parallel execution on generic computers, rather than by reciting a new technology. Improved speed and load balancing are expected results of running a workload in parallel; they are performance outcomes, not inventive technical means. The '582 Patent claims do not improve the internal operation of the computer processors or change how the network functions—they just use more computing power in a straightforward way. Merely accelerating an existing process via parallelization is akin to the scenario in S*AP America v. InvestPic*, where claims "selecting certain information, analyzing it using mathematical techniques, and reporting or displaying the results of the analysis" were held abstract. 898 F.3d

1161, 1167-1168 (Fed. Cir. 2018). Indeed, the claims of the '582 Patent "fit into the familiar class of claims that do not 'focus ... on [] an improvement in computers as tools, but on certain independently abstract ideas that use computers as tools.'" *Id.* at 1168 (quoting *Electric Power*, 830 F.3d at 1354). Furthermore, IV's reliance on *Enfish, LLC v. Microsoft Corp.*, 822 F.3d 1327 (Fed. Cir. 2016), is misplaced. Unlike *Enfish*, where the claims improved database memory structure, the '582 Patent does not specify any new hardware or improvement to system architecture—it merely applies a standard method of parallelization to a computing process.

Because the Asserted Claims of the '582 Patent of the '582 Patent are directed to a fundamental method for organizing and processing tasks, it falls squarely within the category of claims deemed ineligible under Step 1 of the *Alice* framework.

b. The '582 Patent lacks an inventive concept.

As to *Alice* Step 2, The Asserted Claims of the '582 Patent recite only conventional steps and generic components without introducing any meaningful technical improvements or innovations. The claim elements, individually and in combination, reflect routine operations in distributed computing, as explicitly acknowledged in the specification. Partitioning input data into subsets, distributing tasks to processors, and processing tasks in parallel were standard techniques for optimizing workload distribution at the time. Dkt#1-6 at 3:35-4:15, 4:59-63.

The first-come/first-served scheduling technique, likewise, is a well-established method for processing tasks based on availability, like the applicant's example of the four-way stop sign. The Asserted Claims of the '582 Patent "merely appl[y] [this] well-known idea using generic computers." *Symantec*, 838 F.3d at 1314. The specification does not describe any modification, improvement, or adaptation of this scheduling technique to make it unique or unconventional in the context of distributed computing. The Asserted Claims of the '582 Patent simply apply this known concept predictably to allocate unprocessed subtasks, a principle embedded in all multi-

threaded and distributed computing architectures, and offers no specific mechanisms or technical enhancements. Thus, The Asserted Claims of the '582 Patent are "directed to a result or effect that itself is the abstract idea and merely invoke[s] generic processes and machinery" rather than "a specific means or method that improves the relevant technology." *Yu v. Apple Inc.*, 1 F.4th 1040, 1043 (Fed. Cir. 2021). "What is claimed is simply a generic environment in which to carry out the abstract idea." *Id.*

Confirming the routine nature of the claimed methods, both the specification and the Asserted Claims of the '582 Patent rely entirely on generic computing components, such as processors, memory, and standard network architecture. Dkt#1-6 at 4:27–36; *Trinity Info Media, LLC v. Covalent, Inc.*, 72 F.4th 1355, 1364, 1366–67 (Fed. Cir. 2023) (generic computer components such as "processors" and "memory" insufficient to add an inventive concept). The patent does not propose any novel configurations or functionalities for these components but instead recites conventional "functions in general terms, without limiting them to technical means for performing the functions that are arguably an advance over conventional computer and network technology." *Elec. Power*, 830 F.3d at 1351.

For example, Claim 1's reliance on functional language—such as "automatically determining," "logically subdividing," "distributing," "executing," and "combining"—describes desired outcomes without specifying how those outcomes are achieved. It does not explain how data is partitioned into subsets, how subtasks are allocated, how unallocated tasks are managed, or how results are aggregated into a final output. Dkt#1-6 at 6:47–4:15, 4:59–63. This lack of specificity leaves the claim framed at a high level of abstraction, with no concrete implementation details that could transform the method into a patent-eligible application. The dependent Asserted Claims of the '582 Patent fare no better. Claim 2 merely delegates partitioning and distribution to

another processor, without explaining how that processor performs the task. Claim 5 generically allows output in arbitrary order but lacks any technical detail on how that's achieved. Claim 7 limits the input file to network-attached storage—a conventional environment the patent admits was already well known.

Courts have consistently held analogous claims ineligible under Step 2 of the Alice framework. In *Content Extraction*, 776 F.3d at 1348, the court found "no 'inventive concept'" in the use of generic computer components to perform routine activities. Similarly, in *Symantec Corp.*, 838 F.3d at 1320, the Federal Circuit emphasized that for a computer-implemented invention to be patent-eligible, it must involve more than well-understood, routine, and conventional activities. Like the claims in these cases, the Asserted Claims of the '582 Patent merely apply abstract principles of task management to distributed computing using conventional techniques, which does not suffice to render it patent-eligible.

IV points to the patent's description of using heterogeneous computers and balancing load as evidence that the claims capture a technical innovation. This argument is unpersuasive. As detailed above, the combination of steps in the Asserted Claims of the '582 Patent are entirely conventional—they are the standard recipe for parallel processing (split, distribute, process, repeat, merge). There is no inventive coordination or unconventional synergy arising from the arrangement of these steps; each step does what one would normally expect, and together they achieve the expected outcome of parallel execution. Unlike in *CosmoKey*, where the claim recited a specific authentication method that was not routine, or *Enfish* where a new data structure improved computer functionality, the Asserted Claims of the '582 Patent simply assemble known techniques to carry out an abstract goal. 15 F.4th at 1098-1099; 822 F.3d at 1337. Combining conventional steps in a computer environment, without more, does not create an inventive concept.

IV emphasizes that the patent describes using multiple heterogeneous computers and optimizing execution, but those aspects were already familiar in distributed computing. *See* Section III.F.1 *supra*.

Viewed as a whole, the Asserted Claims of the '582 Patent combine conventional steps in a predictable manner to achieve the abstract goal of managing distributed tasks. They offer no technological advancements or inventive concepts beyond the abstract idea itself. Accordingly, the '582 Patent fails to meet the requirements for patent eligibility under 35 U.S.C. § 101.

IV. THE '844 PATENT IS INVALID UNDER 35 U.S.C. §§ 102 AND/OR 103

A. The '844 Patent Priority Date

The asserted claims of the '844 Patent have a filing date of February 21, 2007. Application No. 11/709,477 (the '477 Application), which matured into the '844 Patent, was filed on February 21, 2007 and was a continuation-in-part of Application No. 11/395,816 (filed March 30, 2006) (now Patent No. 7,721,282) (the '816 Application), which was a continuation-in-part of Application No. 11/026,622 (filed December 20, 2004) (now abandoned) (the '622 Application). Plaintiff's Infringement Contentions allege that the asserted claims of the '844 Patent are entitled to a priority date of December 30, 2004, which is the filing date of the '622 Application.

However, none of the '844 Patent claims are entitled to either priority date, as the '816 and '622 Applications never mention caching or indexing – at least one of which is expressly recited in each claim of the '844 Patent. '844 Patent, 10:55- 14:17. *See Studiengesellschaft Kohle v. Shell Oil Co.*, 112 F.3d 1561, 1564 (Fed. Cir. 1997). Applicants even admitted that "indexing and indexing results are not discussed anywhere within Nguyen" (which matured from a continuation-in-part of the '622 Application). *See* '477 Application, Amendment After Non-Final Rejection dated February 21, 2007. Therefore, the asserted claims of the '844 Patent are only entitled to a filing date of February 21, 2007.

B. Identification of Prior Art that Anticipates or Renders Obvious the Asserted Claims of the '844 Patent

Southwest contends that the prior art references listed below and charted in Exhibits 844-1 – 844-24 anticipate and/or render obvious, alone or in combination, one or more of the asserted claims of the '844 Patent.

Prior Art Reference	Publication Date
"Optimizing the Migration of Virtual Computers," Association for Computing Machinery, Special Interest Group on Operating Systems, Operating Systems Review, Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Volume 36, Issue S1 ("Sapuntzakis")	Published December 31, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
U.S. Patent No. 7,089,300 ("Birse")	Filed – October 19, 1999 Issued – August 8, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2005/0283597 ("Holzmann")	Filed – June 22, 2004 Published – December 22, 2005 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni")	Filed – January 8, 2001 Published – March 21, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
U.S. Patent Application Publication No. 2005/0004925 ("Stahl")	Filed – May 7, 2004 Published – January 6, 2005 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).

Prior Art Reference	Publication Date
U.S. Patent Application Publication No. 2006/0168154 ("Zhang")	Filed – November 19, 2004 Published – July 27, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 8,001,085 ("Kiselev")	Filed – November 25, 2003 Issued – August 16, 2011 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2004/0172509 ("Takeda")	Filed – June 23, 2003 Published – September 2, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
U.S. Patent No. 6,671,782 ("Menon")	Filed – March 29, 2000 Issued – December 30, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
U.S. Patent Application Publication No. 2003/0236850 ("Kodama")	Filed – March 21, 2002 Published – December 25, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
EP Patent No. EP0604013 ("Nelson")	Filed – November 15, 1993 Issued – February 20, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
PCT Application Publication No. WO 2006/074869 ("Bayer")	Filed – January 4, 2006 Published – July 20, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
Shrira, Liuba et al., "Thresher: an efficient storage manager for copy-on-write snapshots," ATEC '06: Proceedings of the	Published at least by May 30, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(a).

Prior Art Reference	Publication Date
annual conference on USENIX '06 Annual Technical Conference (Shrira)	
Laros, James H. et al., "Implementing Scalable Disk-Less Clusters Using The Network File System (NFS)," ("Laros")	Published at least by January 2003 Retrieved from: https://www.researchgate.net/publication/255223070_Implementing_scalable_disk-less_clusters_using_the_Network_File_System_NFS This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
U.S. Patent Application Publication No. 2004/0153639 ("Cherian")	Filed – February 5, 2003 Published – August 5, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent Application Publication No. 2007/0057958 ("Bucher")	Filed – August 9, 2006 Published – March 15, 2007 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 7,197,608 ("Mikuma")	Filed – July 12, 2004 Issued – March 27, 2007 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2005/0283575 ("Kobayashi")	Filed – August 13, 2004 Published – December 22, 2005 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent No. 6,453,334 ("Vinson")	Filed – June 16, 1998 Issued – September 17, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent No. 5,930,513 ("Taylor")	Filed – June 6, 1996

Prior Art Reference	Publication Date
	Issued – July 27, 1999 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent Application Publication No. 2002/0002660 ("Malcolm")	Filed – July 31, 1998 Published – January 3, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent Application Publication No. 2002/0091763 ("Shah")	Filed – May 15, 2001 Published – July 11, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent No. 7,398,382 ("Rothman")	Filed – December 29, 2004 Issued – July 8, 2008 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 6,317,826 ("McCall")	Filed – October 19, 1998 Issued – November 13, 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
Southwest Airlines NetApp System – NetApp/Southwest	At least as early as 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) or (g).
VMware Systems - VMware	At least as early as 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) or (g).

Plaintiffs assert claims 7 and 11 of the '844 Patent against Southwest in this lawsuit. These claims are invalid because the '844 Patent fails to meet one or more of the requirements for patentability. The individual bases for invalidity are provided below and in Exhibits 844-1 – 844-

24. Each of the prior art references, the underlying work, and/or the underlying apparatus or method qualifies as prior art under one or more sections of Pre-AIA 35 U.S.C. §§ 102 and/or 103.

In an effort to focus the issues, Southwest has cited representative portions of identified references, even where a reference may contain additional disclosures or support for a particular claim element. Each and every disclosure of the same limitation in the same reference is not necessarily identified. In addition, persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature and in the context of their knowledge as persons of ordinary skill in the art. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other information within the reference, along with other publications and their general scientific knowledge and experience in the field. Thus, each prior art reference identified herein should be understood in the context of the reference itself in conjunction with anything incorporated by reference or described as relevant technology by the reference, any documents that are referred to for further understanding by the reference, any product embodying the reference, and any testimony by those with knowledge of the reference, such as the named authors and inventors. Southwest may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited. Southwest may also rely on uncited portions of the prior art references, other disclosed publications, and the testimony of experts to establish that a person of ordinary skill in the art would have been motivated to modify or combine certain of the cited references so as to render the claims obvious.

Some or all of the asserted claims of the ‘844 Patent are invalid as anticipated under Pre-AIA 35 U.S.C. § 102 in view of certain ones of the prior art references identified above as detailed

in the claim charts included in Exhibits 844-1 – 844-24, which identify specific examples of where each limitation of the asserted claims of the '844 Patent is found in the prior art references. As explained above, the cited portions of prior art references identified in the attached claim charts are exemplary in nature and representative of the content and teaching of the prior art references, and should be understood in the context of the reference as a whole and as they would be understood by a person of ordinary skill in the art.

To the extent any item of prior art cited above is deemed not to disclose, explicitly or inherently, any limitation of an asserted claim of the '844 Patent, Southwest reserves the right to argue that any difference between that prior art and the corresponding patent claim would have been either inherent in the art or obvious to a person of ordinary skill in the art, even if Southwest has not specifically denoted that the art is to be combined with the knowledge of a person of ordinary skill in the art.

To the extent Plaintiffs argue that any limitation of the asserted claims of the '844 Patent is not explicitly disclosed by an item of prior art identified above and in Exhibits 844-1 – 844-24, any purported differences are such that the claimed subject matter as a whole would have been obvious to one skilled in the art at the time of the alleged invention, in view of the state of the art and knowledge of those skilled in the art. The item of prior art would, therefore, render the relevant claims invalid as obvious under Pre-AIA 35 U.S.C. § 103.

To the extent any of the foregoing references are found not to anticipate the asserted claims of the '844 Patent, the foregoing references render the asserted claims obvious, either alone or in combination with one or more of the other references identified above. One of ordinary skill in the art would have been motivated to combine the various references cited herein so as to disclose each and every limitation of the asserted claims of the '844 Patent.

The prior art identified in Exhibits 844-1 – 844-24 renders one or more asserted claims of the '844 Patent obvious when the references are read alone or in combination with one or more of each other and/or when read in view of the state of the art and knowledge of those skilled in the art.

To the extent that Plaintiff argues that any asserted claim of the '844 Patent is not disclosed or rendered obvious by any reference charted in Exhibits 844-1 – 844-24, the asserted claims nevertheless would have been obvious in light of one or more other references charted in Exhibits 844-1 – 844-24 and in view of the knowledge, experience, and understanding of persons of ordinary skill in the art.

Each and every reference identified is also relevant to the state of the art at the time of the alleged invention. Any of the references disclosed above may be combined with one another to render obvious (and therefore invalid) each of the asserted claims of the '844 Patent. Southwest may rely upon a subset of the above identified references, or all of the references identified above, including all references in charted in Exhibits 844-1 – 844-24, for purposes of obviousness depending on the Court's claim construction, positions taken by Plaintiffs during this litigation, and further investigation and discovery.

1. Example Combinations of the Prior Art

The table below provides an identification of exemplary combinations of prior art references that, based on Southwest's present understanding of Plaintiff's Infringement Contentions and apparent reading of the claims, render obvious one or more of the asserted claims of the '844 Patent. The combinations of references identified below are non-exhaustive, and additional examples are apparent from other combinations of references charted in Exhibits 844-1 – 844-24. Although certain references charted in Exhibits 844-1 – 844-24 are listed variously in the left or right column below, Southwest does not thereby imply any restriction as to which

reference is primary and which reference is secondary in such a combination. Southwest reserves the right to rely on either identified reference in the identified combinations as a primary and/or secondary reference.

Exhibit	Reference	Combination Reference(s)
844-1	Sapuntzakis	Holzmann, Nelson, Stahl, and/or Kulkarni, and/or one or more references as charted in Exhibits Exhibits 844-1 – 844-24
844-2	Birse	Holzmann, Nelson, Stahl, and/or Kulkarni, and/or one or more references as charted in Exhibits Exhibits 844-1 – 844-24
844-3	Kulkarni	Holzmann, Nelson, and/or Stahl, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-4	Stahl	Holzmann, Nelson, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits Exhibits 844-1 – 844-24
844-5	Zhang	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-6	Kiselev	Holzmann, Nelson, and/or Stahl, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-7	Takeda	Holzmann, Nelson, and/or Stahl, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-8	Menon	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-9	Kodama	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or

Exhibit	Reference	Combination Reference(s)
		more references as charted in Exhibits 844-1 – 844-24
844-10	Laros	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-11	Shrira	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-12	Bayer	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-13	Cherian	One or more references as charted in Exhibits 844-1 – 844-24
844-14	Bucher	One or more references as charted in Exhibits 844-1 – 844-24
844-15	Mikuma	One or more references as charted in Exhibits 844-1 – 844-24
844-16	Kobayashi	One or more references as charted in Exhibits 844-1 – 844-24
844-17	Vinson	One or more references as charted in Exhibits 844-1 – 844-24
844-18	Taylor	One or more references as charted in Exhibits 844-1 – 844-24
844-19	Malcolm	One or more references as charted in Exhibits 844-1 – 844-24
844-20	Shah	One or more references as charted in Exhibits 844-1 – 844-24
844-21	Rothman	One or more references as charted in Exhibits 844-1 – 844-24

Exhibit	Reference	Combination Reference(s)
844-22	McCall	One or more references as charted in Exhibits 844-1 – 844-24
844-23	Southwest Airlines NetApp System	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits 844-1 – 844-24
844-24	VMware Systems	Holzmann, Nelson, Stahl, Sapuntzakis, Birse, and/or Kulkarni, and/or one or more references as charted in Exhibits 844-1 – 844-24

It will be understood that, given the similarities between the art, any of Sapuntzakis, Birse, Kulkarni, Stahl, Zhang, Kiselev, Takeda, Menon, Kodama, Holzmann, Nelson, Cherian, Bucher, Mikuma, Kobayashi, Vinson, Taylor, Malcolm, Shah, Rothman, McCall, Laros, Shrira, Bayer, Southwest Airlines NetApp System, and/or VMware Systems could be combined with one or more of each other, for example based on the disclosures shown in Exhibits 844-1 – 844-24. It would have been obvious to combine these references for reasons that are readily apparent to one of ordinary skill in the art, and for reasons explained in additional detail in the sections that follow.

2. The Claims Recite Well-Known Concepts that Would Have Been Obvious to Implement

As noted above, the asserted claims of the ‘844 Patent represent nothing more than an arrangement of well-known concepts and features and add nothing to the state of the art at the time of the alleged invention. For each of the concepts represented by the claims, it would have been inherently obvious to add to the teachings of any number of known references and products, including to any of the primary references charted in Exhibits 844-1 – 844-24 to the extent that such reference is alleged to be missing such concept or feature.

3. Additional Reasons to Combine

In addition to the information contained elsewhere in these invalidity contentions, including in Exhibits 844-1 – 844-24, Southwest has identified additional motivations and reasons to combine the various references. Southwest notes that in *KSR Int'l Co. v. Teleflex Inc.*, the Supreme Court held that "[i]n determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." 550 U.S. 398, 127 S.Ct. 1727, 1742 (2007).

KSR further illustrated several ways in which the subject matter of a patent claim may be shown to be obvious. For instance, "[o]ne of the ways in which a patent's subject matter can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *KSR*, 127 S.Ct. 1727, 1742. The Supreme Court held that it was error to "look only to the problem the patentee was trying to solve." *Id*. Rather, "[t]he question is not whether the combination was obvious to the patentee but whether the combination was obvious to a person with ordinary skill in the art. Under the correct analysis, any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id*. Further, "[c]ommon sense teaches … that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* Further, a showing that a combination of elements was "obvious to try" may show that it was obvious under § 103. *Id.* For instance, the Supreme Court held that "predictable solutions" using a combination of "known options" may render the subject matter of a patent claim obvious.

When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. *Id.* The ways in which the subject matter of a patent claim may be shown to be obvious, identified by *KSR*, are merely illustrative. The main thrust of *KSR* was that "[r]igid preventative rules that deny factfinders recourse to common sense"—such as the overturned "teaching, suggestion, or motivation" test—are neither necessary under our case law nor consistent with it." *KSR*, 127 S.Ct. 1727, 1742-43; citing with approval, e.g., *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F. 3d 1356, 1367 (2006) ("Our suggestion test is in actuality quite flexible and not only permits, but requires, consideration of common knowledge and common sense"); *Alza Corp. v. Mylan Labs., Inc.*, 464 F. 3d 1286, 1291 (2006) ("There is flexibility in our obviousness jurisprudence because a motivation may be found implicitly in the prior art. We do not have a rigid test that requires an actual teaching to combine...").

Although Southwest notes that there is no longer a rigid requirement regarding motivation to combine under *KSR*, Southwest provides the following statements regarding motivation to combine. Multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 844-1 – 844-24 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because

known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one.

The combination of prior art references identified in these contentions would have been obvious because the combination represents the known potential options with a reasonable expectation of success. Additionally, one of ordinary skill in the art would have been motivated to create combinations identified in these contentions using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or teaching, suggestion, or motivation in the prior art generally. Also, market forces in the industry, and the desire to improve features and performance, would motivate the addition of features to products as they become available, become less expensive, become more commonly used, provide better performance, reduce costs, or predictably achieve other clearly desirable results.

Moreover, in addition to the information contained elsewhere in these contentions, and, in particular, charted in Exhibits 844-1 – 844-24, Southwest has identified additional motivations and reasons to combine the various references cited herein. In particular, multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 844-1 – 844-24 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market

forces either in the same field or a different one. The combination of prior art references identified in these contentions would have been obvious because the claimed combinations represent the known potential options with a reasonable expectation of success.

To the extent that Plaintiffs contend that any of the above-identified references does not expressly or inherently disclose a particular element or limitation, it would have been obvious to a person of ordinary skill in the art to modify the references or to combine their teachings with other prior art references, including the prior art references in these invalidity contentions, in a manner that renders such claims obvious and invalid.

In particular, various references identified above disclose, well before the critical date of the '844 Patent, systems distributed across plurality of compute nodes that provide for creating cohesive application environments by tracking changes to a common image. For example, Sapuntzakis concerns a system architecture for quickly moving the state of a running computer (including its OS, applications and user files on disk, and all data in memory) – called a "capsule" – across a network to another machine. *See* Sapuntazakis, 2. This is achieved by reducing the amount of data sent over the network through use of, for instance, "copy-on-write" ("COW") disk images that track just the updates to the disk-storage component of capsules. *Id*.

As another example, Birse relates to the provision of a OS in a net-booted environment. *See* Birse, 1:18-21. According to one embodiment, a server in a network computer ("NC") system maintains a copy of the OS that cannot be corrupted by ordinary users of the NC system. Additionally, the NC system may preserve user customizations (e.g., preferences) by maintaining individual, user, storage areas. When an NC client boots from the network and accesses a stored copy of the OS from an NC server, the user's preferences are dynamically merged with the system

environment provided to the NC client. *Id.* 2:56-65. Advantageously, because of these features, a user may login to any NC client on the network and have the same user experience. *Id.* 2:65-3:3.

As yet another example, Kulkarni discloses a system comprising a server and a personal digital assistant (PDA). The server is equipped with a processor, memory, and a network interface, enabling communication with remote devices over a network, such as the Internet. The server's memory contains an image. *See* Kulkarni, at [0027]. The image in flash memory serves as the starting point for creating an updated image. A differences file is stored that includes new data sections, data not present in the root image, and references to unchanged sections of the root image. *See* Kulkarni at [0038].

As yet another example, Stahl discloses a read-only repository file system 226 and a shared read-only file system that provides data to multiple computers. It further describes an overlay file system 220, which stores copies of files that have been modified by respective computers. Stahl also discloses a repository FS instance 224 and MapFS instance 216, which serve as a cache configured to store transient, in-memory versions of cached data blocks previously accessed by compute nodes, enabling efficient data access. *See* Stahl at [0026]-[0027].

As yet another example, Zhang describes a data module that stores primary data objects on a computer storage medium. It further discloses replication and placement of data objects in a distributed system, where replicas are created that may include only modified or additional data blocks specific to the respective compute nodes. Zhang at [0034], [0038]. Additionally, Zhang describes an index module on compute nodes that builds and tracks indices of data objects and shares this information with other nodes, enabling efficient access and management of data across the distributed system. *See* Zhang at [0008], [0047].

As yet another example, Kiselev discloses a system for maintaining consistent and up-to-date copies of primary data across a primary node and a secondary node. The primary node stores the primary data, while the secondary node holds snapshots that capture changes made to the primary data using overlay objects. These overlay objects store only the modified data blocks, referencing unchanged blocks from the primary volume. *See* Kiselev at Abstract, 5:26-63; 6:28-63.

As yet another example, Takeda discloses a data processing system comprising a primary storage unit and multiple secondary storage units. The primary storage unit (PVOL 2212) serves as the main data volume, storing data blocks used for updates and journal acquisition. The secondary storage units (SVOL 2214) store journal data and metadata derived from updates made to the primary storage unit. The journal data includes information about updates, such as addresses, timestamps, and content changes. The secondary storage unit (SVOL 2214) does not duplicate unchanged data blocks from the primary storage unit, as journal data represents only changes and additions. Additionally, the storage controller (210) incorporates a cache memory (212) to enhance I/O performance by temporarily storing data blocks. *See* Takeda at [0035]-[0038].

As yet another example, Menon discloses a shared disk system where multiple nodes, each equipped with a computer and an adaptor, interface with shared storage devices. The shared storage devices (10a, b, c) act as the primary repository storing data blocks used by compute nodes. Each adaptor includes a write cache (20a, b, c), which temporarily holds updates or modifications made by individual compute nodes. Additionally, the adaptors feature read caches (18a, b, c) that temporarily store recently accessed data blocks, optimizing data retrieval and enhancing I/O performance for compute nodes. *See* Menon at 3:55-4:61.

As yet another example, Kodama discloses a system where a primary storage unit stores a master copy of data. Kodama explains that replicated volumes store copies of the data from the master volume. The system handles changes to the data, such as modifications and deletions, by writing them to previously unused areas of the storage, with the updated data being replicated to the replicated volumes. *See* Kodama at [0007], [0050]-[0053].

As yet another example, the NetApp system implemented at Southwest Airlines in 2002 exemplifies a robust and efficient method for managing root and leaf image data. The system architecture at Southwest leveraged clustered NetApp filers integrated with Oracle9i RAC to achieve high availability and seamless data management. The key technologies used in this setup included WAFL (Write Anywhere File Layout), SnapMirror, and SnapRestore. These technologies facilitated the management of root and leaf image data, maintained consistent application environments, and supported rapid recovery after system disruptions.

As yet another example, the VMware Systems, introduced in the early 2000s, enabled multiple virtual machines (compute nodes) to run concurrently on a single physical server. The virtual machines share data through root image files stored on centralized storage units and manage data changes via snapshots and redo logs. The systems efficiently provide data to a plurality of compute nodes through virtualization and dynamic resource allocation. *See* VMware GSX Server 3 Virtual Machine Guide, VMware GSX Server 3 Specifications Datasheet, and VMware ESX Server 2.1 Specifications Datasheet.

Various references identified above also disclose, well before the critical date of the ‘844 Patent, caching of data blocks. For example, Holzmann is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. *See* Holzmann, [0017]. When a boot request is received from an external device, the control

module 103 allocates cache space in the volatile cache memory portion 104 of the disk. The cache is used in the typical manner to optimize future access to data once it has demonstrated some likelihood of being further referenced. As additional servers or other network resources are booted, the cache is used to speed access to the most commonly used data. *Id.* at Abstract, [0021], [0024].

As another example, Nelson discloses a method and system for a caching file server. It stores the data in the cache memory accessed by the first client. A second client does not need to access the data directly from the storage device, the data can be directly accessed by the cache memory. *See* Nelson at [0038]-[0039].

A person of ordinary skill in the art would have been motivated to combine the above references to incorporate the different beneficial technical features provided by each of the above references. Modifying the above references would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results.

* * *

In addition to the specific combinations of prior art and the specific combinations of groups of prior art disclosed above, Southwest reserves the right to rely on any other combination of any prior art references disclosed herein. Southwest further reserves the right to rely upon combinations of prior art references disclosed within the prosecution history of the references cited herein.

The references charted in Exhibits 844-1 – 844-24 disclose the elements of the asserted claims of the '844 Patent explicitly and/or inherently and may also be relied upon to show the state of the art at the time of the alleged invention. The suggested obviousness combinations are provided in the alternative to Southwest's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not itself anticipatory.

C. OGP: Charts Setting Forth Where in the Prior Art References Each Element of the Asserted Claims Are Found

Pursuant to the Court's OGP, charts setting forth where specifically in each alleged item of prior art each element of each Asserted Claim of the '844 Patent is found are attached as Exhibits 844-1 – 844-24.

D. Other Relevant Prior Art References

In addition to the references identified above and charted in Exhibits 844-1 – 844-24, each of the references below is relevant to the '844 Patent, as it relates to the state of the art. Southwest reserves the right to amend its contentions to rely on any of the below references alone or in combination with any other references identified in these contentions.

Inventor/Author	Country	Issue/Publication No.	Filed Date
C. K. Haun et al.	US	2004/0153526	8/5/2004
James E. King et al.	US	7,529,807	5/5/2004
Yevgeniy Klimenko et al.	US	5,974,547	3/20/1998
Beng-Hong Lim	US	6,795,966	2/4/2000
Julian S. Taylor	US	5,930,513	6/6/1996
Tung M. Nguyen	US	7,870,106	2/2/2006
Pradip Kulkarni	US	7,721,282	5/18/2010
Rauch, et al., "Partition Repositories for Partition Cloning — OS Independent Software Maintenance in Large Clusters of PCs," Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER 2000), DOI: 10.1109/CLUSTR.2000.889066			Pub. November 28, 2000.
Goscinski, et al., "GENESIS: An Efficient, Transparent and Easy to Use Cluster Operating System," *Parallel Computing*, Vol. 28, Issue 4, Pages 557–606			Pub. at least as early as April 2002

V. THE '722 PATENT IS INVALID UNDER 35 U.S.C. §§ 102 AND/OR 103

A. The '722 Patent Priority Date

Application No. 11/396,251 (the '251 Application), which matured into the '722 Patent, was filed on March 30, 2006 and was a continuation of Application No. 10/105,018 (filed March 21, 2002) (now Patent No. 7,051,070) Application No. 10/105,018 is a continuation-in-part of Application No. 10/017,182 (filed Dec. 14, 2001) (now Patent No. 7,043,525).

The '722 Patent further claims the benefit of several provisional applications: Provisional Application No. 60/256,613 (filed Dec. 18, 2000); Provisional Application No. 60/276,847 (filed March 16, 2001); Provisional Application No. 60/278,303 (filed March 21, 2001); Provisional Application No. 60/279,608 (filed March 27, 2001); and Provisional Application No. 60/280,627 (filed March 29, 2001).

Plaintiff's Infringement Contentions allege that the asserted claims of the '722 Patent are entitled to a priority date of December 18, 2000, which is the filing date of the '613 Provisional. However, only one of the provisional applications, the '303 Provisional, filed on March 21, 2001, describes message types and node categories, which are claimed elements in the '722 Patent. Therefore, the asserted claims of the '722 Patent are only entitled to a filing date of March 21, 2001.

B. Identification of Prior Art that Anticipates or Renders Obvious the Asserted Claims of the '722 Patent

Southwest contends that the prior art references listed below and charted in Exhibits 722-1 – 722-11 anticipate and/or render obvious, alone or in combination, one or more of the asserted claims of the '722 Patent.

Prior Art Reference	Publication Date
U.S. Patent No. 5,721,825 ("Lawson")	Filed October 3, 1996 Issued – February 24, 1998 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
U.S. Patent No. 6,480,883 ("Tsutsumitake")	Filed – June 29, 1999 Issued – November 12, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
PCT Patent Application Publication No. WO 99/26127 ("Kostes")	Filed – November 13, 1998 Published – May 27, 1999 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
U.S. Patent No. 6,999,991 ("Ikeda")	Filed – July 24, 2000 Issued – February 14, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 6,990,591 ("Pearson")	Filed – December 22, 1999 Issued – January 24, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
European Patent Application Publication No. EP1043671 ("Bird")	Filed – March 17, 2000 Published – October 11, 2000 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(a).
U.S. Patent No. 6,721,740 ("Skinner")	Filed – June 5, 1998 Issued – April 13, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 6,710,702 ("Averbuch")	Filed – November 22, 1999 Issued – March 23, 2004

Prior Art Reference	Publication Date
	This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 6,763,384 ("Gupta")	Filed – July 10, 2000 Issued – July 13, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 6,859,829 ("Parupudi")	Filed – February 23, 1999 Issued – February 22, 2005 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 6,216,132 ("Chandra")	Filed – November 20, 1997 Issued – April 10, 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 6,910,070 ("Mishra")	Filed – January 24, 2000 Issued – June 21, 2005 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 6,823,359 ("Heidingsfeld")	Filed – November 21, 2000 Issued – November 23, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 6,757,283 ("Yamanaka")	Filed – January 24, 2000 Issued – June 29, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 7,051,084 ("Hayton")	Filed – November 2, 2000 Issued – May 23, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).

Prior Art Reference	Publication Date
United States Patent No. 6,633,910 ("Rajan")	Filed – December 14, 1999 Issued – October 14, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
United States Patent No. 9,300,560 ("Leighton")	Filed – March 4, 2013 Priority - July 20, 2000 Issued – March 29, 2016 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
PointCast System – PointCast Incorporated	At least as early as 1997 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) or (g).
Castanet System – Marimba, Inc.	At least as early as 1996 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) or (g).
Active Desktop System – Microsoft Corporation	At least as early as 1998 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) or (g).

Plaintiffs assert claims 14, 16, 17, and 18 of the '722 Patent against Southwest in this lawsuit. These claims are invalid because the '722 Patent fails to meet one or more of the requirements for patentability. The individual bases for invalidity are provided below and in Exhibits 722-1 - 722-11. Each of the prior art references, the underlying work, and/or the underlying apparatus or method qualifies as prior art under one or more sections of Pre-AIA 35 U.S.C. §§ 102 and/or 103.

In an effort to focus the issues, Southwest has cited representative portions of identified references, even where a reference may contain additional disclosures or support for a particular claim element. Each and every disclosure of the same limitation in the same reference is not

necessarily identified. In addition, persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature and in the context of their knowledge as persons of ordinary skill in the art. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other information within the reference, along with other publications and their general scientific knowledge and experience in the field. Thus, each prior art reference identified herein should be understood in the context of the reference itself in conjunction with anything incorporated by reference or described as relevant technology by the reference, any documents that are referred to for further understanding by the reference, any product embodying the reference, and any testimony by those with knowledge of the reference, such as the named authors and inventors. Southwest may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited. Southwest may also rely on uncited portions of the prior art references, other disclosed publications, and the testimony of experts to establish that a person of ordinary skill in the art would have been motivated to modify or combine certain of the cited references so as to render the claims obvious.

Some or all of the asserted claims of the ‘722 Patent are invalid as anticipated under Pre-AIA 35 U.S.C. § 102 in view of certain ones of the prior art references identified above as detailed in the claim charts included in Exhibits 722-1 - 722-11, which identify specific examples of where each limitation of the asserted claims of the ‘722 Patent is found in the prior art references. As explained above, the cited portions of prior art references identified in the attached claim charts are exemplary in nature and representative of the content and teaching of the prior art references,

and should be understood in the context of the reference as a whole and as they would be understood by a person of ordinary skill in the art.

To the extent any item of prior art cited above is deemed not to disclose, explicitly or inherently, any limitation of an asserted claim of the '722 Patent, Southwest reserves the right to argue that any difference between that prior art and the corresponding patent claim would have been either inherent in the art or obvious to a person of ordinary skill in the art, even if Southwest has not specifically denoted that the art is to be combined with the knowledge of a person of ordinary skill in the art.

To the extent Plaintiffs argue that any limitation of the asserted claims of the '722 Patent is not explicitly disclosed by an item of prior art identified above and in Exhibits 722-1 - 722-11, any purported differences are such that the claimed subject matter as a whole would have been obvious to one skilled in the art at the time of the alleged invention, in view of the state of the art and knowledge of those skilled in the art. The item of prior art would, therefore, render the relevant claims invalid as obvious under Pre-AIA 35 U.S.C. § 103.

To the extent any of the foregoing references are found not to anticipate the asserted claims of the '722 Patent, the foregoing references render the asserted claims obvious, either alone or in combination with one or more of the other references identified above. One of ordinary skill in the art would have been motivated to combine the various references cited herein so as to disclose each and every limitation of the asserted claims of the '722 Patent.

The prior art identified in Exhibits 722-1 - 722-11 renders one or more asserted claims of the '722 Patent obvious when the references are read alone or in combination with one or more of each other and/or when read in view of the state of the art and knowledge of those skilled in the art.

To the extent that Plaintiff argues that any asserted claim of the '722 Patent is not disclosed or rendered obvious by any reference charted in Exhibits 722-1 - 722-11, the asserted claims nevertheless would have been obvious in light of one or more other references charted in Exhibits 722-1 - 722-11 and in view of the knowledge, experience, and understanding of persons of ordinary skill in the art.

Each and every reference identified is also relevant to the state of the art at the time of the alleged invention. Any of the references disclosed above may be combined with one another to render obvious (and therefore invalid) each of the asserted claims of the '722 Patent. Southwest may rely upon a subset of the above identified references, or all of the references identified above, including all references in charted in Exhibits 722-1 - 722-11, for purposes of obviousness depending on the Court's claim construction, positions taken by Plaintiffs during this litigation, and further investigation and discovery.

1. Example Combinations of the Prior Art

The table below provides an identification of exemplary combinations of prior art references that, based on Southwest's present understanding of Plaintiff's Infringement Contentions and apparent reading of the claims, render obvious one or more of the asserted claims of the '722 Patent. The combinations of references identified below are non-exhaustive, and additional examples are apparent from other combinations of references charted in Exhibits 722-1 - 722-11. Although certain references charted in Exhibits 722-1 - 722-11 are listed variously in the left or right column below, Southwest does not thereby imply any restriction as to which reference is primary and which reference is secondary in such a combination. Southwest reserves the right to rely on either identified reference in the identified combinations as a primary and/or secondary reference.

Exhibit	Reference	Combination Reference(s)
722-1	Lawson	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, Pearson, PointCast System, Castanet System, and/or Active Desktop System, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-2	Kostes	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, Pearson, PointCast System, Castanet System, and/or Active Desktop System, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-3	Ikeda	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, Pearson, PointCast System, Castanet System, and/or Active Desktop System, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-4	Skinner	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-5	Gupta	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-6	Parapudi	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-7	Heidingsfeld	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-8	Leighton	Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or

Exhibit	Reference	Combination Reference(s)
		Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-9	PointCast System	Lawson, Kostes, Ikeda, Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-10	Castanet System	Lawson, Kostes, Ikeda, Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11
722-11	Active Desktop System	Lawson, Kostes, Ikeda, Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, and/or Pearson, and/or one or more references as charted in Exhibits 722-1 - 722-11

It will be understood that, given the similarities between the art, any of Lawson, Kostes, Ikeda, Skinner, Gupta, Parapudi, Heidingsfeld, Averbuch, Chandra, Mishra, Yamanaka, Hayton, Rajan, Tsutsumitake, Bird, Pearson, Leighton, Castanet System, PointCast System, and/or Active Desktop System could be combined with one or more of each other, for example based on the disclosures shown in Exhibits 722-1 - 722-11. It would have been obvious to combine these references for reasons that are readily apparent to one of ordinary skill in the art, and for reasons explained in additional detail in the sections that follow.

2. The Claims Recite Well-Known Concepts that Would Have Been Obvious to Implement

As noted above, the asserted claims of the '722 Patent represent nothing more than an arrangement of well-known concepts and features and add nothing to the state of the art at the time of the alleged invention. For each of the concepts represented by the claims, it would have been

inherently obvious to add to the teachings of any number of known references and products, including to any of the primary references charted in Exhibits 722-1 - 722-11 to the extent that such reference is alleged to be missing such concept or feature.

3. Additional Reasons to Combine

In addition to the information contained elsewhere in these invalidity contentions, including in Exhibits 722-1 - 722-11, Southwest has identified additional motivations and reasons to combine the various references. Southwest notes that in *KSR Int'l Co. v. Teleflex Inc.*, the Supreme Court held that "[i]n determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." 550 U.S. 398, 127 S.Ct. 1727, 1742 (2007).

KSR further illustrated several ways in which the subject matter of a patent claim may be shown to be obvious. For instance, "[o]ne of the ways in which a patent's subject matter can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *KSR*, 127 S.Ct. 1727, 1742. The Supreme Court held that it was error to "look only to the problem the patentee was trying to solve." *Id*. Rather, "[t]he question is not whether the combination was obvious to the patentee but whether the combination was obvious to a person with ordinary skill in the art. Under the correct analysis, any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id*. Further, "[c]ommon sense teaches … that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* Further, a showing that a combination of elements was "obvious to try" may show that it was obvious under § 103. *Id.* For instance, the Supreme

Court held that "predictable solutions" using a combination of "known options" may render the subject matter of a patent claim obvious.

When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. *Id.* The ways in which the subject matter of a patent claim may be shown to be obvious, identified by *KSR*, are merely illustrative. The main thrust of *KSR* was that "[r]igid preventative rules that deny factfinders recourse to common sense"—such as the overturned "teaching, suggestion, or motivation" test—are neither necessary under our case law nor consistent with it." *KSR*, 127 S.Ct. 1727, 1742-43; citing with approval, e.g., *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F. 3d 1356, 1367 (2006) ("Our suggestion test is in actuality quite flexible and not only permits, but requires, consideration of common knowledge and common sense"); *Alza Corp. v. Mylan Labs., Inc.*, 464 F. 3d 1286, 1291 (2006) ("There is flexibility in our obviousness jurisprudence because a motivation may be found implicitly in the prior art. We do not have a rigid test that requires an actual teaching to combine...").

Although Southwest notes that there is no longer a rigid requirement regarding motivation to combine under *KSR*, Southwest provides the following statements regarding motivation to combine. Multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 722-1 - 722-11 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one

of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one.

The combination of prior art references identified in these contentions would have been obvious because the combination represents the known potential options with a reasonable expectation of success. Additionally, one of ordinary skill in the art would have been motivated to create combinations identified in these contentions using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or teaching, suggestion, or motivation in the prior art generally. Also, market forces in the industry, and the desire to improve features and performance, would motivate the addition of features to products as they become available, become less expensive, become more commonly used, provide better performance, reduce costs, or predictably achieve other clearly desirable results.

Moreover, in addition to the information contained elsewhere in these contentions, and, in particular, charted in Exhibits 722-1 - 722-11, Southwest has identified additional motivations and reasons to combine the various references cited herein. In particular, multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 722-1 - 722-11 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above

because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one. The combination of prior art references identified in these contentions would have been obvious because the claimed combinations represent the known potential options with a reasonable expectation of success.

To the extent that Plaintiffs contend that any of the above-identified references does not expressly or inherently disclose a particular element or limitation, it would have been obvious to a person of ordinary skill in the art to modify the references or to combine their teachings with other prior art references, including the prior art references in these invalidity contentions, in a manner that renders such claims obvious and invalid.

In particular, various references identified above disclose, well before the critical date of the '722 Patent, systems for providing live objects and routing updates to live objects via various nodes based on message and node types to clients registered to receive updates to the live objects.

For example, Ikeda discloses a push service system that routes data to registered client devices through various agents (nodes) depending on the type of node. See Ikeda at Abstract, 19:48-20:20, 20:28-47.

As another example, Lawson discloses "[a] system [and] method for global event notification in a distributed computer environment" in which "a local event registry [is used] to identify local event consumers that should be notified when an event occurs" and "[A] global event registry [] identifies other servers which need notification when an event occurs." These other servers "will then, in turn, notify their local event consumers of the event." *See* Lawson at Abstract.

As yet another example, Kostes, discloses "a system and method for displaying, in near real-time, data published from various computer systems and networks, such as the Internet." 1:22-

24. The system is "web-enabled, providing tight integration with all popular [web] browsers." 30:17-20. In Kostes's "publish/subscribe architecture [(aka ''model-view' architecture')] ... various information sources [(e.g., different sources such as Sportsticker, CNN, Reuters, etc.)], most likely under a contractual agreement with [Kostes's] push server, constantly publish and broadcast data to the push server, using any desired communication method," and a "client computer subscribes to desired types of information available from the push server." 7:12-18, 16:10-13, 2:11-20, 3:9-14.

As yet another example, Tsutsumitake discloses "[a] real-time information transmission system [and method that] can immediately transmit information updated on a server to a client" in a web-enabled environment. *See* Tsutsumitake at Abstract; 1:6-32. A web "client analyzes [a web]page. If an event request is included in the analyzed page ... [a] connection ID [is used by] the server [to] send[] an event to the client each time the event has occurred. The client processes the event and reflects the processed result on a screen of a display" to, e.g., update in real time powerplant data 71 and 72 being monitored via the web. *See* Tsutsumitake at Abstract, 13:31-65.

As yet another example, the PointCast System provided a push/subscribe service in which live webpage objects such as news, financial data, and other content was dynamically updated within a graphical UI. *See* https://web.archive.org/web/19990826121133/http://www.pointcast.com/tour/index.html?hometxt. The PointCast server acted as a gateway, receiving news or stock updates from external sources and associating them with specific content channels. The PointCast server/gateway identified the category of update message and then sent the update message to the appropriate node type. *See* https://www.pcmag.com/encyclopedia/term/pointcast;

https://web.archive.org/web/19990921040538/http://www.pointcast.com/content/index.html?hometxt.

As yet another example, the Castanet System, introduced in 1996, was a family of Java-based delivery systems for publishing and automatically distributing application updates and other published materials via the Internet and intranets." *See* https://www.pcmag.com/encyclopedia/term/marimba.

As yet another example, the Active Desktop System is a Microsoft Windows feature since Windows 95 that allows users to display interactive content, such as web pages, as their desktop wallpaper. *See* https://gustavus.edu/gts/Active_Desktop. Active desktop "provid[ed] a dynamic and personalized computing experience, transforming your static desktop into an interactive hub. By embedding live web content like news feeds or weather updates directly onto your background, it keeps you informed without the need to open multiple applications. This feature enhances convenience and productivity, offering real-time information briefly. Tailoring your desktop with active desktop ensures a workspace that aligns with your preferences, making it a valuable tool for those seeking a more engaging and customized digital environment." *See* https://www.lenovo.com/us/en/glossary/active-desktops/.

Various references identified above also disclose, well before the critical date of the '722 Patent, identifying a message category based on an input source. For example, Bird discloses a publish/subscribe system in which messages are identified according to the sender. *See* Bird at Abstract, [0044]. As another example, Pearson discloses including source identifying information in message headers such as port numbers. *See* Pearson at 18:41-19:18.

A person of ordinary skill in the art would have been motivated to combine the above references to incorporate the different beneficial technical features provided by each of the above

references. Modifying the above references would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results.

* * *

In addition to the specific combinations of prior art and the specific combinations of groups of prior art disclosed above, Southwest reserves the right to rely on any other combination of any prior art references disclosed herein. Southwest further reserves the right to rely upon combinations of prior art references disclosed within the prosecution history of the references cited herein.

The references charted in Exhibits 722-1 - 722-11 disclose the elements of the asserted claims of the '722 Patent explicitly and/or inherently and may also be relied upon to show the state of the art at the time of the alleged invention. The suggested obviousness combinations are provided in the alternative to Southwest's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not itself anticipatory.

C. OGP: Charts Setting Forth Where in the Prior Art References Each Element of the Asserted Claims Are Found

Pursuant to the Court's OGP, charts setting forth where specifically in each alleged item of prior art each element of each Asserted Claim of the '722 Patent is found are attached as Exhibits 722-1 - 722-11.

D. Additional Prior Art

In addition to the references identified above and charted in Exhibits 722-1 - 722-11, each of the references below is relevant to the '722 Patent, as it relates to the state of the art. Southwest reserves the right to amend its contentions to rely on any of the below references alone or in combination with any other references identified in these contentions.

Inventor/Author	Country	Issue/Publication No.	Filed Date
Robert W. Baynes Jr. et al.	US	2002/0032722	9/12/2001
Simon Antony James Holdsworth et al.	US	7,103,680	4/25/2000
Andrew John Mark Miles et al.	US	6,886,044	4/26/2005
Mark A. Fox et al.	US	6,654,786	8/30/2000
Richard Lee Schwartz et al.	US	5,913,032	9/30/1996
Hemant Madan	US	7,516,196	3/21/2000
Christopher C. Andrews	US	2010/0325562	3/1/2000
Stephen James Paul Todd	US	6,643,682	2/22/2000
Hirokawa Masaru et al.	JP	JPH11353216A	06/08/1998; Pub. 12/24/1999
James P. Richardson et al.	US	6,292,803	11/18/1997
Colin Leonard Bird et al.	UK	GB 2 348 025	3/19/1999; Pub 9/20/2000
M. Hauswirth and M. Jazayeri, "A Component and Communication Model for Push Systems," Proceedings of the ESEC/FSE 99 - Joint 7th European Software Engineering Conference ("Hauswirth")			Pub. at least as early as September 6, 1999.
G. S. Barnes Nelson, "Messaging Systems: SAS® Tools for Internet-Based Communications," Proceedings of the 25th Annual SAS Users Group International Conference (SUGI 25), Paper 288-25 ("Barnes Nelson")			Pub. at least as early as March 19, 1999.
Segall, et al., "Content Based Routing with Elvin4," Proceedings of AUUG2K (No. 39) ("Segall")			Pub. at least as early as June 2000.

VI. THE '785 PATENT IS INVALID UNDER 35 U.S.C. §§ 102 AND/OR 103

A. The '785 Patent Priority Date

Plaintiff's Infringement Contentions allege the asserted claims of the '785 Patent have a filing date of March 21, 2003. However, this appears to be a typographical error. The '785 Patent has a filing date of March 31, 2003. The '785 Patent includes no priority claim.

B. Identification of Prior Art that Anticipates or Renders Obvious the Asserted Claims of the '785 Patent

Southwest contends that the prior art references listed below and charted in Exhibits 785-1 – 785-29 anticipate and/or render obvious, alone or in combination, one or more of the asserted claims of the '785 Patent.

Prior Art Reference	Publication Date
U.S. Patent No. 6,970,941 ("Caronni I")	Filed – December 10, 1999 Issued – November 29, 2005 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 7,814,228 ("Caronni II")	Filed – February 13, 2003 Issued – October 12, 2010 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 6,766,371 ("Hipp")	Filed – October 5, 2000 Issued – July 20, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2003/0028671 ("Mehta")	Filed – June 10, 2002 Published – February 6, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
Huitema, C., Network Working Group Request for Comment (RFC): 1383, entitled An Experiment in DNS Based IP Routing ("RFC 1383")	Published December 1992 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
U.S. Patent No. 7,133,404 ("Alkhatib")	Filed – August 10, 2001 Issued – November 7, 2006 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No.	Filed – April 19, 2002 Published – December 5, 2002

Prior Art Reference	Publication Date
2002/0184529 (“Foster I”)	This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2003/0208602 (“Bhalla”)	Filed – April 8, 2002 Published – November 6, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0181395 (“Foster II”)	Filed – April 19, 2002 Published – December 5, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0091859 (“Tuomenoksa”)	Filed – April 11, 2001 Published – July 11, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2004/0177136 (“Chen”)	Filed – March 3, 2003 Published – September 9, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
European Patent Publication No. EP1098496 (“Engmann”)	Published – October 19, 2000 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
U.S. Patent Application Publication No. 2002/0069278 (“Forslow”)	Filed – December 5, 2000 Published – June 6, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 6,701,437 (“Hoke”)	Filed – November 9, 1998 Issued – March 2, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0116502 (“Iyer”)	Filed – February 22, 2001 Published – August 22, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).

Prior Art Reference	Publication Date
U.S. Patent Application Publication No. 2003/0065785 ("Jain")	Filed – September 28, 2001 Published – April 3, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0035624 ("Kim")	Filed – July 9, 2001 Published – March 21, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent Application Publication No. 20020154635 ("Liu")	Filed – April 23, 2001 Published – October 24, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 8,068,817 ("Viswanath")	Filed – August 27, 2002 Issued – November 29, 2011 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0116523 ("Warrier")	Filed – February 22, 2001 Published – August 22, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2003/0229697 ("Borella")	Filed – June 10, 2002 Published – December 11, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2003/0084162 ("Johnson")	Filed – October 31, 2001 Published – May 1, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
PCT Patent Application Publication No. WO2003088625 ("Volz")	Filed – March 28, 2003 Priority – April 8, 2002 Published – October 23, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No.	Filed – November 20, 2002 Published – May 20, 2004

Prior Art Reference	Publication Date
2004/0098507 ("Thubert")	This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2003/0016636 ("Tari")	Filed – July 15, 2002 Published – January 23, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 8,107,483 ("Dunk")	Filed – March 12, 2010 Priority – November 27, 2002 Issued – January 31, 2012 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
PCT Patent Application Publication No. WO2002015014 ("Wootton")	Filed – July 31, 2001 Published – February 21, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent Application Publication No. 2002/0038382 ("Ryu")	Filed – August 30, 2001 Published – March 28, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e).
U.S. Patent Application Publication No. 2003/0172184 ("Kong")	Filed – March 6, 2003 Published – September 11, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2004/0037260 ("Kakemizu")	Filed – August 7, 2003 Published – February 26, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
Cisco VPN Routers and Software Systems – Cisco Systems	At least as early as 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) and (g).
VMware Systems - VMware	At least as early as 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) and (g).

Plaintiffs assert claims 30, 35, and 37 of the '785 Patent against Southwest in this lawsuit. These claims are invalid because the '785 Patent fails to meet one or more of the requirements for patentability. The individual bases for invalidity are provided below and in Exhibits 785-1 – 785-29. Each of the prior art references, the underlying work, and/or the underlying apparatus or method qualifies as prior art under one or more sections of Pre-AIA 35 U.S.C. §§ 102 and/or 103.

In an effort to focus the issues, Southwest has cited representative portions of identified references, even where a reference may contain additional disclosures or support for a particular claim element. Each and every disclosure of the same limitation in the same reference is not necessarily identified. In addition, persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature and in the context of their knowledge as persons of ordinary skill in the art. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other information within the reference, along with other publications and their general scientific knowledge and experience in the field. Thus, each prior art reference identified herein should be understood in the context of the reference itself in conjunction with anything incorporated by reference or described as relevant technology by the reference, any documents that are referred to for further understanding by the reference, any product embodying the reference, and any testimony by those with knowledge of the reference, such as the named authors and inventors. Southwest may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited. Southwest may also rely on uncited portions of the prior art references, other disclosed publications, and the testimony of experts to establish that a person of ordinary skill in the art

would have been motivated to modify or combine certain of the cited references so as to render the claims obvious.

Some or all of the asserted claims of the '785 Patent are invalid as anticipated under Pre-AIA 35 U.S.C. § 102 in view of certain ones of the prior art references identified above as detailed in the claim charts included in Exhibits 785-1 – 785-29, which identify specific examples of where each limitation of the asserted claims of the '785 Patent is found in the prior art references. As explained above, the cited portions of prior art references identified in the attached claim charts are exemplary in nature and representative of the content and teaching of the prior art references, and should be understood in the context of the reference as a whole and as they would be understood by a person of ordinary skill in the art.

To the extent any item of prior art cited above is deemed not to disclose, explicitly or inherently, any limitation of an asserted claim of the '785 Patent, Southwest reserves the right to argue that any difference between that prior art and the corresponding patent claim would have been either inherent in the art or obvious to a person of ordinary skill in the art, even if Southwest has not specifically denoted that the art is to be combined with the knowledge of a person of ordinary skill in the art.

To the extent Plaintiffs argue that any limitation of the asserted claims of the '785 Patent is not explicitly disclosed by an item of prior art identified above and in Exhibits 785-1 – 785-29, any purported differences are such that the claimed subject matter as a whole would have been obvious to one skilled in the art at the time of the alleged invention, in view of the state of the art and knowledge of those skilled in the art. The item of prior art would, therefore, render the relevant claims invalid as obvious under Pre-AIA 35 U.S.C. § 103.

To the extent any of the foregoing references are found not to anticipate the asserted claims of the '785 Patent, the foregoing references render the asserted claims obvious, either alone or in combination with one or more of the other references identified above. One of ordinary skill in the art would have been motivated to combine the various references cited herein so as to disclose each and every limitation of the asserted claims of the '785 Patent.

The prior art identified in Exhibits 785-1 – 785-29 renders one or more asserted claims of the '785 Patent obvious when the references are read alone or in combination with one or more of each other and/or when read in view of the state of the art and knowledge of those skilled in the art.

To the extent that Plaintiff argues that any asserted claim of the '785 Patent is not disclosed or rendered obvious by any reference charted in Exhibits 785-1 – 785-29, the asserted claims nevertheless would have been obvious in light of one or more other references charted in Exhibits 785-1 – 785-29 and in view of the knowledge, experience, and understanding of persons of ordinary skill in the art.

Each and every reference identified is also relevant to the state of the art at the time of the alleged invention. Any of the references disclosed above may be combined with one another to render obvious (and therefore invalid) each of the asserted claims of the '785 Patent. Southwest may rely upon a subset of the above identified references, or all of the references identified above, including all references in charted in Exhibits 785-1 – 785-29, for purposes of obviousness depending on the Court's claim construction, positions taken by Plaintiffs during this litigation, and further investigation and discovery.

1. Example Combinations of the Prior Art

The table below provides an identification of exemplary combinations of prior art references that, based on Southwest's present understanding of Plaintiff's Infringement

Contentions and apparent reading of the claims, render obvious one or more of the asserted claims of the '785 Patent. The combinations of references identified below are non-exhaustive, and additional examples are apparent from other combinations of references charted in Exhibits 785-1 – 785-29. Although certain references charted in Exhibits 785-1 – 785-29 are listed variously in the left or right column below, Southwest does not thereby imply any restriction as to which reference is primary and which reference is secondary in such a combination. Southwest reserves the right to rely on either identified reference in the identified combinations as a primary and/or secondary reference.

Exhibit	Reference	Combination Reference(s)
785-1	Mehta	RFC 1383, Caronni I, Caronni II, and/or Hipp, and/or one or more references as charted in Exhibits 785-1 – 785-29
785-2	Caronni I	Caronni II, Hipp, Mehta, and/or RFC 1383, and/or one or more references as charted in Exhibits 785-1 – 785-29
785-3	Alkhatib	One or more references as charted in Exhibits 785-1 – 785-29
785-4	Foster I	One or more references as charted in Exhibits 785-1 – 785-29
785-5	Bhalla	One or more references as charted in Exhibits 785-1 – 785-29
785-6	Foster II	One or more references as charted in Exhibits 785-1 – 785-29
785-7	Tuomenoksa	One or more references as charted in Exhibits 785-1 – 785-29
785-8	Chen	One or more references as charted in Exhibits 785-1 – 785-29
785-9	Engmann	One or more references as charted in Exhibits 785-1 – 785-29

Exhibit	Reference	Combination Reference(s)
785-10	Forslow	One or more references as charted in Exhibits 785-1 – 785-29
785-11	Hoke	One or more references as charted in Exhibits 785-1 – 785-29
785-12	Iyer	One or more references as charted in Exhibits 785-1 – 785-29
785-13	Jain	One or more references as charted in Exhibits 785-1 – 785-29
785-14	Kim	One or more references as charted in Exhibits 785-1 – 785-29
785-15	Liu	One or more references as charted in Exhibits 785-1 – 785-29
785-16	Viswanath	One or more references as charted in Exhibits 785-1 – 785-29
785-17	Warrier	One or more references as charted in Exhibits 785-1 – 785-29
785-18	Borella	One or more references as charted in Exhibits 785-1 – 785-29
785-19	Johnson	One or more references as charted in Exhibits 785-1 – 785-29
785-20	Volz	One or more references as charted in Exhibits 785-1 – 785-29
785-21	Thubert	One or more references as charted in Exhibits 785-1 – 785-29
785-22	Tari	One or more references as charted in Exhibits 785-1 – 785-29
785-23	Dunk	One or more references as charted in Exhibits 785-1 – 785-29
785-24	Wootton	One or more references as charted in Exhibits 785-1 – 785-29

Exhibit	Reference	Combination Reference(s)
785-25	Ryu	One or more references as charted in Exhibits 785-1 – 785-29
785-26	Kong	One or more references as charted in Exhibits 785-1 – 785-29
785-27	Kakemizu	One or more references as charted in Exhibits 785-1 – 785-29
785-28	Cisco VPN Routers and Software Systems	Mehta, RFC 1383, Caronni I, Caronni II, and/or Hipp, and/or one or more references as charted in Exhibits 785-1 – 785-29
785-29	VMware Systems	Mehta, RFC 1383, Caronni I, Caronni II, and/or Hipp, and/or one or more references as charted in Exhibits 785-1 – 785-29

It will be understood that, given the similarities between the art, any of Mehta, Caronni I, Caronni II, Hipp, RFC 1383, Alkhatib, Foster I , Bhalla, Foster II, Tuomenoksa, Chen, Engmann, Forslow, Hoke, Iyer, Jain, Kim, Liu, Viswanath, Warrier, Borella, Johnson, Volz, Thubert, Tari, Dunk, Wootton, Ryu, Kong, Kakemizu, Cisco VPN Routers and Software Systems, and/or VMware Systems could be combined with one or more of each other, for example based on the disclosures shown in Exhibits 785-1 – 785-29. It would have been obvious to combine these references for reasons that are readily apparent to one of ordinary skill in the art, and for reasons explained in additional detail in the sections that follow.

2. The Claims Recite Well-Known Concepts that Would Have Been Obvious to Implement

As noted above, the asserted claims of the ‘785 Patent represent nothing more than an arrangement of well-known concepts and features and add nothing to the state of the art at the time of the alleged invention. For each of the concepts represented by the claims, it would have been

inherently obvious to add to the teachings of any number of known references and products, including to any of the primary references charted in Exhibits 785-1 – 785-29 to the extent that such reference is alleged to be missing such concept or feature.

3. Additional Reasons to Combine

In addition to the information contained elsewhere in these invalidity contentions, including in Exhibits 785-1 – 785-29, Southwest has identified additional motivations and reasons to combine the various references. Southwest notes that in *KSR Int'l Co. v. Teleflex Inc.*, the Supreme Court held that "[i]n determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." 550 U.S. 398, 127 S.Ct. 1727, 1742 (2007).

KSR further illustrated several ways in which the subject matter of a patent claim may be shown to be obvious. For instance, "[o]ne of the ways in which a patent's subject matter can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *KSR*, 127 S.Ct. 1727, 1742. The Supreme Court held that it was error to "look only to the problem the patentee was trying to solve." *Id*. Rather, "[t]he question is not whether the combination was obvious to the patentee but whether the combination was obvious to a person with ordinary skill in the art. Under the correct analysis, any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id*. Further, "[c]ommon sense teaches … that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* Further, a showing that a combination of elements was "obvious to try" may show that it was obvious under § 103. *Id.* For instance, the Supreme

Court held that "predictable solutions" using a combination of "known options" may render the subject matter of a patent claim obvious.

When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. *Id.* The ways in which the subject matter of a patent claim may be shown to be obvious, identified by *KSR*, are merely illustrative. The main thrust of *KSR* was that "[r]igid preventative rules that deny factfinders recourse to common sense"—such as the overturned "teaching, suggestion, or motivation" test—are neither necessary under our case law nor consistent with it." *KSR*, 127 S.Ct. 1727, 1742-43; citing with approval, e.g., *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F. 3d 1356, 1367 (2006) ("Our suggestion test is in actuality quite flexible and not only permits, but requires, consideration of common knowledge and common sense"); *Alza Corp. v. Mylan Labs., Inc.*, 464 F. 3d 1286, 1291 (2006) ("There is flexibility in our obviousness jurisprudence because a motivation may be found implicitly in the prior art. We do not have a rigid test that requires an actual teaching to combine...").

Although Southwest notes that there is no longer a rigid requirement regarding motivation to combine under *KSR*, Southwest provides the following statements regarding motivation to combine. Multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 785-1 – 785-29 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one

of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one.

The combination of prior art references identified in these contentions would have been obvious because the combination represents the known potential options with a reasonable expectation of success. Additionally, one of ordinary skill in the art would have been motivated to create combinations identified in these contentions using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or teaching, suggestion, or motivation in the prior art generally. Also, market forces in the industry, and the desire to improve features and performance, would motivate the addition of features to products as they become available, become less expensive, become more commonly used, provide better performance, reduce costs, or predictably achieve other clearly desirable results.

Moreover, in addition to the information contained elsewhere in these contentions, and, in particular, charted in Exhibits 785-1 – 785-29, Southwest has identified additional motivations and reasons to combine the various references cited herein. In particular, multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 785-1 – 785-29 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above

because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one. The combination of prior art references identified in these contentions would have been obvious because the claimed combinations represent the known potential options with a reasonable expectation of success.

To the extent that Plaintiffs contend that any of the above-identified references does not expressly or inherently disclose a particular element or limitation, it would have been obvious to a person of ordinary skill in the art to modify the references or to combine their teachings with other prior art references, including the prior art references in these invalidity contentions, in a manner that renders such claims obvious and invalid.

In particular, various references identified above disclose, well before the critical date of the '785 Patent, virtual network managers having a network interface and that communicate via a virtual network defined by a domain name having a public network address, and that include a register module to register devices in the virtual network.

For example, Mehta discloses a system designed to enable secure, bi-directional communication between public network devices and wireless devices on private networks. *See* Mehta at Abstract. Mehta discloses an Address Management Proxy System (AMPS) that allocates temporary public IP addresses to wireless devices to facilitate communication across public and private networks without exposing private IP addresses. Mehta at Abstract. Mehta's system includes modified DNS/API servers, Address Proxy/Routers, and an Address Management Data Server to dynamically assign public IP addresses to private network devices as needed. *See* Mehta at [0011]. Metha's AMPS uses techniques like Time to Live (TTL) for each IP mapping, ensuring temporary mappings that enhance security by automatically expiring connections. *See* Mehta at

[0015]. The system manages a pool of public addresses to allocate dynamically, using protocols like TCP/IP and UDP for seamless data exchange. *See* Mehta at [0011], [0024]. AMPS can support various types of networks and devices beyond traditional IP networks, including ATM networks and other wireless technologies. Metha addresses the challenge of secure and efficient communication between disparate network types, enhancing connectivity options without compromising network security. *See* Mehta at [0041].

As another example, RFC 1383 discloses "a scheme that allows for simple routing" that is "complementary with the classic hierarchical routing approach," while providing "an easy to implement and low cost solution for 'multi-homed' domains." *See* RFC 1383 at 2. Specifically, the RFC-1383 solution is described as "a generalization of the 'MX record' scheme used for mail routing." *See* RFC 1383 at 2. RFC-1383 introduces "RX records" for routing IP packets by associating domain names with preferred gateways by means of "source routing" or "tunneling," or another form of encapsulation protocol. *See* RFC 1383 at 4. To implement this RX routing, a modified DNS lookup is suggested: the source or an early relay makes the routing decision by querying the DNS for RX records, which list gateways with preference levels. *See* RFC 1383 at 6, 9. RFC-1383 describes a network interface integrated with a standard IP router, working alongside a DNS query manager. *See* RFC 1383 at 12-13. When an Internet Control Message Protocol (ICMP) message is received, the query manager updates the local routing table to ensure that any new packets bound for the specified destination are routed through the real-time forwarder. *Id.* Simultaneously, the query manager sends a DNS request to read the RX records for the destination. *Id.* Upon receiving the response, the query manager selects a gateway and provides this information to the real-time forwarder. *See* RFC 1383 at 11-12. When the real-time forwarder receives a packet, it checks if a gateway for the destination is available. *See* RFC 1383 at 12-13. If

so, it inserts the necessary source routing information and forwards the packet—either through the general IP routing program or directly to the network interface linked to the intermediate gateway. *See* RFC 1383 at 12.

As another example, Caronni I discloses "data processing systems and, more particularly, [] a private network using a public-network infrastructure." *See* Caronni I at 1:57–59. Caronni I establishes a "'Supernet,' which is a private network that uses components from a public-network infrastructure." *See* Caronni I at 4:38–39. The Supernet allows organizations to utilize public networks for their enterprise networks, reducing the burden on maintaining network infrastructure. *See* Caronni I at 4:40–43. Caronni I's system uses a separate layer that isolates address names of nodes from addressing schemes and delivery schemes. *See* Caronni I at 4:55–59 ("The Supernet contains a modification to the IP packet format that can be used to separate network behavior from addressing. As a result of the modification, any delivery scheme may be assigned to any address, or group of addresses."). This separation provides flexibility in addressing and delivery methods. When a node attempts to join a Supernet through the SNlogin Script, SASD validates and authenticates the request and then "creates an address mapping between a node ID and the real address." *See* Caronni I at 9:67–10:18 (SNlogin script "performs a handshaking with SASD to authenticate this information.").

Caronni II discloses "a virtual computer network and more particularly to the addressing of messages sent in a virtual network using data encapsulated in higher level protocols." *See* Caronni II at 1:6–9. In particular, Caronni II addresses the challenge of sending messages to virtual addresses in networks with packet filtering or network address translation (NAT) devices. *See* Caronni II at 3:54–57, 8:14–25. Caronni II introduces an extended virtual address registration process that includes higher-level protocol designations, allowing messages to bypass filtering

mechanisms. *See* Caronni II at 2:40–44. Caronni II discloses a virtual network overlay on a physical network, where "[a] virtual network is a logical network overlaid on a physical network." *See* Caronni II at 1:13–14. Caronni II utilizes a virtual address resolution facility that "works with stored virtual address registrations for the virtual network." *See* Caronni II at 4:9–10. These registrations are stored in a Virtual Address Resolution Protocol (VARP) lookup table and include not only the virtual IP address corresponding real IP address but also "a designation of a transport protocol and a port address for an interface between the Application layer and the Transport layer." *See* Caronni II at 4:62–65.

As another example, Hipp discloses "[a] virtual network environment to be used by a set of applications for the express purpose of isolating the applications from other applications on the same node or network[.]" *See* Hipp at Abstract. The "Virtual Network Environment" (VNE) is defined as "a collection of IP addresses related to applications that are contained within the[] VNE or have the potential of being placed in the VNE." *See* Hipp at 2:66–3:2. The system assigns a "unique virtual IP address and virtual hostname to a group of processes that make up the application instance which the instance keeps throughout its execution." *See* Hipp at 6:1–4. This virtual network identity (VNI) remains with the application regardless of which node it runs on, allowing for transparent movement across machines without modifying the application. *See* Hipp at 6:4–6.

Further, Hipp demonstrates that using a domain name server (DNS) was well-known and understood well before the '785 Patent. Hipp discloses using a DNS in the context of hostname resolution. *See* Hipp at 6:16–25. "The virtual hostname resolves to the virtual IP address for both the applications registered with the VNI framework as well as those that are not registered. This may require configuration of a name service or OS host configuration files." *See* Hipp at 6:16–20.

"For example, if an application instance used a virtual IP address of 10.10.0.1 and a virtual hostname of host1055, the standard hostname to IP address resolution mechanisms (e.g. DNS or the /etc/hosts file) would have to be preconfigured to resolve a query of host1055 to IP address 10.10.0.1." *See* Hipp at 6:20–25.

As yet another example, the Cisco VPN Routers and Software Systems provided devise and software enabling fast data communication through network interfaces, facilitated adding and removing devices from the network, including supporting VPN tunnels for such devices, and performing address translation. *See, e.g.*, Cisco 7100 Series VPN Routers, Cisco VPN 3000 Series Concentrator Data Sheet, Cisco VPN 3000 Series Concentrator Data Sheet.

As yet another example, the VMware Systems provided a virtual infrastructure that provided dynamic resource allocation, enabling dynamic management of server resources and provisioning through a management console, and provisioning of new resources. *See, e.g.*, VMware ESX Server Specifications, VMware GSX Server Specifications, VMware Enterprises Continue to Embrace VMware Virtual Infrastructure.

A person of ordinary skill in the art would have been motivated to combine the above references to incorporate the different beneficial technical features provided by each of the above references. Modifying the above references would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results.

* * *

In addition to the specific combinations of prior art and the specific combinations of groups of prior art disclosed above, Southwest reserves the right to rely on any other combination of any prior art references disclosed herein. Southwest further reserves the right to rely upon

combinations of prior art references disclosed within the prosecution history of the references cited herein.

The references charted in Exhibits 785-1 – 785-29 disclose the elements of the asserted claims of the '785 Patent explicitly and/or inherently and may also be relied upon to show the state of the art at the time of the alleged invention. The suggested obviousness combinations are provided in the alternative to Southwest's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not itself anticipatory.

C. OGP: Charts Setting Forth Where in the Prior Art References Each Element of the Asserted Claims Are Found

Pursuant to the Court's OGP, charts setting forth where specifically in each alleged item of prior art each element of each Asserted Claim of the '785 Patent is found are attached as Exhibits 785-1 – 785-29.

D. Other Relevant Prior Art References

In addition to the references identified above and charted in Exhibits 785-1 – 785-29, each of the references below is relevant to the '785 Patent, as it relates to the state of the art. Southwest reserves the right to amend its contentions to rely on any of the below references alone or in combination with any other references identified in these contentions.

Inventor/Author	Country	Issue/Publication No.	Filed Date
Hasan S. Alkhatib et al.	US	6,119,171	Sept. 12, 2000
Hasan S. Alkhatib et al.	US	7,139,828	Nov. 21, 2006
Hasan S. Alkhatib et al.	US	2004/0044778	Aug. 30, 2002
Jan Forslöw	US	7,155,518	January 8, 2001
Barrett Morris Kreiner et al.	US	2008/0232295	November 27, 2002
Anthony J. Li et al.	US	5,473,599	December 5, 1995
Changwen Lui et al.	US	2004/0120295	December 19, 2002
Samir Mehta et al.	WO	2002/102012A2	June 10, 2002
Emily L. Hipp	US	7,146,431	Dec. 5, 2006
Emily L. Hipp	US	7,210,147	Oct. 5, 2000
Can C. Aysan et al.	US	7,379,465	December 7, 2001

Inventor/Author	Country	Issue/Publication No.	Filed Date
Qiaofeng Zhou	US	2002/0138578	January 24, 2001
Ashar Aziz	US	6,643,701	Nov. 17, 1999
Moon-Jeong Choi	US	2002/0040397	October 2, 2001
Germano Caronni et al.	US	6,798,782	Dec. 10, 1999
Germano Caronni et al.	US	7,685,309	Aug. 11, 2005 – priority Dec. 10, 1999
Germano Caronni et al.	US	7,765,581	Dec. 10, 1999
David Spooner	US	8,743,891	Feb. 24, 2004 – priority Aug. 24, 2001
Qiaofend Zhou	US	6,950,879	Jan. 24, 2001
John Keane et al.	US	6,996,628	April 11, 2001
Robert P. St. Peirre et al.	US	7,254,835	Jan. 4, 2002
Robert P. St. Peirre et al.	US	7,890,633	Feb. 13, 2003
James Russell Godwin	US	7,519,721	April 4, 2008 – priority Jan. 17. 2001
Luis F. Cabrera	US	7,899,047	Aug. 13, 2007 – priority Nov. 27, 2001
Werner Lindemann	US	2002/0010796	Jan. 24, 2002
Thomas T. Nguyen	US	2002/0016926	April 26, 2001
Samuel Bendinelli	US	2002/0029276	April 11, 2001
Jheroen P. Dorenbosch	US	2002/0138622	March 21, 2001
John Andrew Aiken JR.	US	2002/0178268	May 22, 2001
Yuefeng Liu	US	2002/0154635	April 23, 2001
Glenn C. Scott	US	2004/0019633	July 24, 2002
Glen Ceniza	US	2002/0186698	June 12, 2001
Germano Caronni et al.	WO	01/43329	Dec. 8, 2000
Germano Caronni et al.	WO	01/45351	Dec. 8, 2000
Germano Caronni et al.	WO	01/43355	Dec. 11, 2000
Germano Caronni et al.	WO	01/43356	Dec. 11, 2000
Germano Caronni et al.	WO	01/43369	Dec. 11, 2000
Germano Caronni et al.	WO	01/43391	Dec. 11, 2000
Holger Karl et al.	WO	2004/075506	Feb. 16, 2004
M. Skye Poier	CA	2316428	Pub. Feb. 18, 2002
Robert P. St. Peirre et al.	UK	2384667	Pub. July 30, 2003
Microsoft Corporation, "Virtual Private Networking," *Microsoft TechNet – Networking & RAS Technology Center*			Pub. Jan. 17, 2000
C. Perkins, "IP Mobility Support for IPv4," *Request for Comments: 3220*, Internet Engineering Task Force			Jan. 18, 2002

Inventor/Author	Country	Issue/Publication No.	Filed Date
C. Perkins, "Mobile IP," *IEEE Communications Magazine*, IEEE Communications Society			Pub. May 1997
G. Caronni et al., "Virtual Enterprise Networks: The Next Generation of Secure Enterprise Networking," Annual Computer Security Applications Conference (2000)			Pub. 2000
VPN 3000 Series Concentrator Getting Started; Cisco Systems - Https://Web.Archive.Org/Web/20030308052045/Http://Www.Cisco.Com/Application/Pdf/En/Us/Guest/Products/Ps2284/C1692/Ccmigration_09186a00801021e2.Pdf			Dated November 2001; Pub. At least by March 8, 2003
Rekhter, Y. et al., Network Working Group Request for Comment (RFC): 1918, entitled Address Allocation for Private Internets			Pub. Feb. 1996
Fox, B. et al., Network Working Group Request for Comment (RFC): 2685, entitled Virtual Private Networks Identifier			Pub. Sept. 1999
Gleeson, B. et al., Network Working Group Request for Comment (RFC): 2764, entitled A Framework for IP Based Virtual Private Networks			Pub. Feb. 2000
Lim, et al., "Customizable virtual private network service with QoS" *Computer Networks*, Volume 36, Issue 1, Pages 137–151, Elsevier Science B.V.			Pub. July 2001
Okazaki, Naonobu et al., "An Efficient Method for Private Network Management Using IP Address Translation," T.IEE Japan, Vo. 121-C, No. 10, 2001			Pub. 2001

Inventor/Author	Country	Issue/Publication No.	Filed Date
Kato, et al., "Mobile IP Using Private IP Addresses," Proceedings of the Sixth IEEE Symposium on Computers and Communications			Pub. July 2001
Zaccone, et al., "Address reuse in the Internet, adjourning or suspending the adoption of IP next generation?," Proceedings of IEEE International Conference on Networks 2000 (ICON 2000) Networking Trends and Challenges in the New Millenium			Pub. 2000
A. Kara, "Secure Remote Access from Office to Home," *IEEE Communications Magazine*, Vol. 39, Issue 10, pp. 68-72			Pub. Oct. 1, 2001
H. Liu, "Mobile Domain Name System An Alternative for Mobile IP," Proceedings of the 8th Annual International Conference on Communications Systems 2002 (ICCS 2002)			Pub. 2002

VII. THE '326 PATENT IS INVALID UNDER 35 U.S.C. §§ 102 AND/OR 103

A. The '326 Patent Priority Date

Plaintiff's Infringement Contentions allege the asserted claims of the '326 Patent have a filing date of January 12, 2004.

B. Identification of Prior Art that Anticipates or Renders Obvious the Asserted Claims of the '326 Patent

Southwest contends that the prior art references listed below and charted in Exhibits 326-1 – 326-11 anticipate and/or render obvious, alone or in combination, one or more of the asserted claims of the '326 Patent.

Prior Art Reference	Publication Date
U.S. Patent Application Publication No. 2003/022473 ("Yamaura")	Filed – February 20, 2003 Published – December 4, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 6,510,133 ("Uesugi")	Filed – January 21, 1999 Issued – January 21, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
JP Patent Application Publication No. JP2002319917 ("Mori")	Filed – April 24, 2001 Published – October 31, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
JP Patent Application Publication No. JP2003309533 ("Miyoshi")	Filed – April 17, 2002 Published – October 31, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(a).
JP Patent Application Publication No. JPH11113049 ("Koga")	Filed – September 30, 1997 Published – April 23, 1999 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
U.S. Patent No. 7,418,050 ("Gardner")	Filed – February 26, 2003 Issued – August 26, 2008 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
PCT Patent Application Publication No. WO 00/38387 ("Baldemair")	Filed – December 21, 1999 Published – October 10, 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
Thompson, D. E., "Modelling Adjacent Channel Interference in 3G Networks," THE INSTITUTE OF	Published – 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).

Prior Art Reference	Publication Date
ELECTRICAL ENGINEERS (2003) ("Thompson")	
Weste, Neil et al., "VLSI for OFDM," IEEE COMMUNICATIONS MAGAZINE (Oct. 1998) ("Weste")	Published – 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(b) and 102(e).
"WLAN Channel Bonding: Causing Greater Problems Than It Solves" ("Texas Instruments White Paper")	Published – September 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
"WLAN Solutions: TNETW1130 Converged Single-Chip MAC and Baseband Processor for IEEE 802.11 a/b/g" ("TI Baseband Processor for IEEE 802.11")	Published – 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).

Plaintiffs assert claims 1, 4, and 18 of the '326 Patent against Southwest in this lawsuit. These claims are invalid because the '326 Patent fails to meet one or more of the requirements for patentability. The individual bases for invalidity are provided below and in Exhibits 326-1 – 326-11. Each of the prior art references, the underlying work, and/or the underlying apparatus or method qualifies as prior art under one or more sections of Pre-AIA 35 U.S.C. §§ 102 and/or 103.

In an effort to focus the issues, Southwest has cited representative portions of identified references, even where a reference may contain additional disclosures or support for a particular claim element. Each and every disclosure of the same limitation in the same reference is not necessarily identified. In addition, persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature and in the context of their knowledge as persons of ordinary skill in the art. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other

information within the reference, along with other publications and their general scientific knowledge and experience in the field. Thus, each prior art reference identified herein should be understood in the context of the reference itself in conjunction with anything incorporated by reference or described as relevant technology by the reference, any documents that are referred to for further understanding by the reference, any product embodying the reference, and any testimony by those with knowledge of the reference, such as the named authors and inventors. Southwest may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited. Southwest may also rely on uncited portions of the prior art references, other disclosed publications, and the testimony of experts to establish that a person of ordinary skill in the art would have been motivated to modify or combine certain of the cited references so as to render the claims obvious.

Some or all of the asserted claims of the '326 Patent are invalid as anticipated under Pre-AIA 35 U.S.C. § 102 in view of certain ones of the prior art references identified above as detailed in the claim charts included in Exhibits 326-1 – 326-11, which identify specific examples of where each limitation of the asserted claims of the '326 Patent is found in the prior art references. As explained above, the cited portions of prior art references identified in the attached claim charts are exemplary in nature and representative of the content and teaching of the prior art references, and should be understood in the context of the reference as a whole and as they would be understood by a person of ordinary skill in the art.

To the extent any item of prior art cited above is deemed not to disclose, explicitly or inherently, any limitation of an asserted claim of the '326 Patent, Southwest reserves the right to argue that any difference between that prior art and the corresponding patent claim would have

been either inherent in the art or obvious to a person of ordinary skill in the art, even if Southwest has not specifically denoted that the art is to be combined with the knowledge of a person of ordinary skill in the art.

To the extent Plaintiffs argue that any limitation of the asserted claims of the '326 Patent is not explicitly disclosed by an item of prior art identified above and in Exhibits 326-1 – 326-11, any purported differences are such that the claimed subject matter as a whole would have been obvious to one skilled in the art at the time of the alleged invention, in view of the state of the art and knowledge of those skilled in the art. The item of prior art would, therefore, render the relevant claims invalid as obvious under Pre-AIA 35 U.S.C. § 103.

To the extent any of the foregoing references are found not to anticipate the asserted claims of the '326 Patent, the foregoing references render the asserted claims obvious, either alone or in combination with one or more of the other references identified above. One of ordinary skill in the art would have been motivated to combine the various references cited herein so as to disclose each and every limitation of the asserted claims of the '326 Patent.

The prior art identified in Exhibits 326-1 – 326-11 renders one or more asserted claims of the '326 Patent obvious when the references are read alone or in combination with one or more of each other and/or when read in view of the state of the art and knowledge of those skilled in the art.

To the extent that Plaintiff argues that any asserted claim of the '326 Patent is not disclosed or rendered obvious by any reference charted in Exhibits 326-1 – 326-11, the asserted claims nevertheless would have been obvious in light of one or more other references charted in Exhibits 326-1 – 326-11 and in view of the knowledge, experience, and understanding of persons of ordinary skill in the art.

Each and every reference identified is also relevant to the state of the art at the time of the alleged invention. Any of the references disclosed above may be combined with one another to render obvious (and therefore invalid) each of the asserted claims of the '326 Patent. Southwest may rely upon a subset of the above identified references, or all of the references identified above, including all references in charted in Exhibits 326-1 – 326-11, for purposes of obviousness depending on the Court's claim construction, positions taken by Plaintiffs during this litigation, and further investigation and discovery.

1. Example Combinations of the Prior Art

The table below provides an identification of exemplary combinations of prior art references that, based on Southwest's present understanding of Plaintiff's Infringement Contentions and apparent reading of the claims, render obvious one or more of the asserted claims of the '326 Patent. The combinations of references identified below are non-exhaustive, and additional examples are apparent from other combinations of references charted in Exhibits 326-1 – 326-11. Although certain references charted in Exhibits 326-1 – 326-11 are listed variously in the left or right column below, Southwest does not thereby imply any restriction as to which reference is primary and which reference is secondary in such a combination. Southwest reserves the right to rely on either identified reference in the identified combinations as a primary and/or secondary reference.

Exhibit	Reference	Combination Reference(s)
326-1	Baldemair	Gardner, Koga, Miyoshi, Mori, Yamaura, Uesegi, Texas Instruments White Paper, TI Baseband Processor for IEEE 802.11, and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-2	Gardner	Miyoshi, Koga, Mori, Uesugi, TI Baseband Processor for IEEE 802.11,

Exhibit	Reference	Combination Reference(s)
		Yamaura, Texas Instruments White Paper, and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-3	Koga	Gardner, Miyoshi, Mori, Uesugi, Yamaura, TI Baseband Processor for IEEE 802.11, Texas Instruments White Paper, and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-4	Miyoshi	Gardner, Koga, Mori, TI Baseband Processor for IEEE 802.11, Texas Instruments White Paper, Yamaura, Uesugi and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-5	Mori	Gardner, Koga, Miyoshi, Yamaura, Texas Instruments White Paper, TI Baseband Processor for IEEE 802.11, Uesugi and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-6	Texas Instruments White Paper	Gardner, Koga, Miyoshi, Mori, TI Baseband Processor for IEEE 802.11, Uesugi, Yamaura, and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-7	Thompson	Gardner, Koga, Miyoshi, Mori, Uesugi, Yamaura, Texas Instruments White Paper, TI Baseband Processor for IEEE 802.11, and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-8	TI Baseband Processor for IEEE 802.11	Yamaura, Uesugi, Mori, Miyoshi, Koga, Gardner, Texas Instruments White Paper and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-9	Uesugi	Gardner, Koga, Miyoshi, Mori, Texas Instruments White Paper, TI Baseband Processor for IEEE 802.11, Yamaura, and/or one or more references as charted in Exhibits 326-1 – 326-11.

Exhibit	Reference	Combination Reference(s)
326-10	Weste	Gardner, Koga, Yamaura, Mori, Miyoshi, Texas Instruments White Paper, TI Baseband Processor for IEEE 802.11, Uesugi, and/or one or more references as charted in Exhibits 326-1 – 326-11.
326-11	Yamaura	Gardner, Koga, Mori, Miyoshi, Texas Instruments White Paper, TI Baseband Processor for IEEE 802.11, Uesugi, and/or one or more references as charted in Exhibits 326-1 – 326-11

It will be understood that, given the similarities between the art, any of Baldemari, Gardner, Koga, Miyoshi, Mori, Texas Instruments White Paper, Thompson, TI Baseband Processor for IEEE 802.11, Uesugi, Weste, and/or Yamaura could be combined with one or more of each other, for example based on the disclosures shown in Exhibits 326-1 – 326-11. It would have been obvious to combine these references for reasons that are readily apparent to one of ordinary skill in the art, and for reasons explained in additional detail in the sections that follow.

2. The Claims Recite Well-Known Concepts that Would Have Been Obvious to Implement

As noted above, the asserted claims of the ‘326 Patent represent nothing more than an arrangement of well-known concepts and features and add nothing to the state of the art at the time of the alleged invention. For each of the concepts represented by the claims, it would have been inherently obvious to add to the teachings of any number of known references and products, including to any of the primary references charted in Exhibits 326-1 – 326-11 to the extent that such reference is alleged to be missing such concept or feature.

3. Additional Reasons to Combine

In addition to the information contained elsewhere in these invalidity contentions, including in Exhibits 326-1 – 326-11, Southwest has identified additional motivations and reasons to combine the various references. Southwest notes that in *KSR Int'l Co. v. Teleflex Inc.*, the Supreme Court held that "[i]n determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." 550 U.S. 398, 419 (2007).

KSR further illustrated several ways in which the subject matter of a patent claim may be shown to be obvious. For instance, "[o]ne of the ways in which a patent's subject matter can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *KSR*, 127 S.Ct. at 1742. The Supreme Court held that it was an error to "look only to the problem the patentee was trying to solve." *Id*. Rather, "[t]he question is not whether the combination was obvious to the patentee but whether the combination was obvious to a person with ordinary skill in the art. Under the correct analysis, any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id*. Further, "[c]ommon sense teaches … that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* Further, a showing that a combination of elements was "obvious to try" may show that it was obvious under § 103. *Id.* For instance, the Supreme Court held that "predictable solutions" using a combination of "known options" may render the subject matter of a patent claim obvious.

When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. *Id.* The ways in which the subject matter of a patent claim may be shown to be obvious, identified by *KSR*, are merely illustrative. The main thrust of *KSR* was that "[r]igid preventative rules that deny factfinders recourse to common sense"—such as the overturned "teaching, suggestion, or motivation" test—are neither necessary under our case law nor consistent with it." *KSR*, 127 S.Ct. at 1742-43; citing with approval, *e.g.*, *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F. 3d 1356, 1367 (2006) ("Our suggestion test is in actuality quite flexible and not only permits, but requires, consideration of common knowledge and common sense"); *Alza Corp. v. Mylan Labs., Inc.*, 464 F. 3d 1286, 1291 (2006) ("There is flexibility in our obviousness jurisprudence because a motivation may be found implicitly in the prior art. We do not have a rigid test that requires an actual teaching to combine . . .").

Although Southwest notes that there is no longer a rigid requirement regarding motivation to combine under *KSR*, Southwest provides the following statements regarding motivation to combine. Multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 326-1 – 326-11 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because

known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one.

The combination of prior art references identified in these contentions would have been obvious because the combination represents the known potential options with a reasonable expectation of success. Additionally, one of ordinary skill in the art would have been motivated to create combinations identified in these contentions using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or teaching, suggestion, or motivation in the prior art generally. Also, market forces in the industry, and the desire to improve features and performance, would motivate the addition of features to products as they become available, become less expensive, become more commonly used, provide better performance, reduce costs, or predictably achieve other clearly desirable results.

Moreover, in addition to the information contained elsewhere in these contentions, and, in particular, charted in Exhibits 326-1 – 326-11, Southwest has identified additional motivations and reasons to combine the various references cited herein. In particular, multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 326-1 – 326-11 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market

forces either in the same field or a different one. The combination of prior art references identified in these contentions would have been obvious because the claimed combinations represent the known potential options with a reasonable expectation of success.

To the extent that Plaintiffs contends that any of the above-identified references does not expressly or inherently disclose a particular element or limitation, it would have been obvious to a person of ordinary skill in the art to modify the references or to combine their teachings with other prior art references, including the prior art references in these invalidity contentions, in a manner that renders such claims obvious and invalid.

For example, many of the prior art references charted in Exhibits 326-1 – 326-11 share a goal of increasing the data rates and data throughput in a network. Various references identified in these Preliminary Infringement Contentions disclose, well before the alleged critical date of the '326 patent, increased speed in data transmission: "In order to increase the speed of the communication channel, it is necessary to allocate a wide frequency bandwidth for each transmission channel." Mori at [0029].

For example, Yamaura discloses "a convention radio communication system" that "accords to the scheme called HiSWANa," where "the frequencies of the center carrier are 5.17 GHz, 5.19 GHz, 5.21 GHz, and 5.23 GHz, and a signal band of 20 MHz (including a guard band) is assigned to each carrier." Yamaura at [0005]. A person of ordinary skill in the art would understand that using a HiSWANa (high-speed wireless access network) scheme for optimal guard band usage is comparable to increasing data rate and throughput.

As another example, Mori discloses selecting adjacent channels comprised of data subcarriers and separated by guard bands: "[t]he radio access system put into practical use uses a bandwidth of 100 MHz in a 5 GHz band, and arranges four channels having an interval of 20 MHz

in the 5-GHz band, and performs communication using each channel." Mori at [0003]. Further, Mori discloses partially filling the frequency gap, or guard band, between the two adjacent channels by adding data subcarriers into the guard band: " . . . the spectrum shown in h) increases the number of data carriers by allocating subcarrier data to the guard band between the continuous bands B1 and B2." Mori at [0131-32]. Further still, Mori discloses filling the guard band using fast fourier transform or inverse fast fourier transform: "subcarriers other than the pilot signal for transmitting the information signal are based on a complex signal sequence supplied to the IFFT," " . . . [t]he IFFT circuit 14 performs inverse Fourier transform on the frequency domain information relating to the signal point position for each subcarrier supplied for each of the plurality of operation blocks, and obtains the time domain information for each IFFT circuit block." Mori at [0012], [0051].

As another example, Gardner discloses combining a first and second channel using channel bonding: "[v]arious options for IEEE 802.11a extensions that would support higher data rates have been considered. One option is to use multiple channels to transmit the data stream; for example, data could be multiplexed between two 54 Mbps streams occupying two contiguous 20 MHz channels to obtain a rate of 108 Mbps." Gardner at [2:15-26].

As set forth above, the market and industry for the increase in data rates and data throughput in a network was well established and known by one of ordinary skill in the art. A person of ordinary skill would know, for example, to consider any feature of increasing data rates and throughput already implemented by major industry players, including systems such as Texas Instruments Baseband Processor for IEEE 802.11, which contemplates increasing data throughput and channel bonding.

* * *

In addition to the specific combinations of prior art and the specific combinations of groups of prior art disclosed above, Southwest reserves the right to rely on any other combination of any prior art references disclosed herein. Southwest further reserves the right to rely upon combinations of prior art references disclosed within the prosecution history of the references cited herein.

The references charted in Exhibits 326-1 – 326-11 disclose the elements of the asserted claims of the '326 Patent explicitly and/or inherently and may also be relied upon to show the state of the art at the time of the alleged invention. The suggested obviousness combinations are provided in the alternative to Southwest's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not itself anticipatory.

C. OGP: Charts Setting Forth Where in the Prior Art References Each Element of the Asserted Claims Are Found

Pursuant to the Court's OGP, charts setting forth where specifically in each alleged item of prior art each element of each Asserted Claim of the '326 Patent is found are attached as Exhibits 326-1 – 326-11.

D. Other Relevant Prior Art References

In addition to the references identified above and charted in Exhibits 326-1 – 326-11, each of the references below is relevant to the '326 Patent, as it relates to the state of the art. Southwest reserves the right to amend its contentions to rely on any of the below references alone or in combination with any other references identified in these contentions.

Inventor/Author	Country	Issue/Publication No.	Filed Date
IEEE STANDARD FOR INFORMATION TECHNOLOGY. ANSI/ IEEE Std 802.11, 1999 Edition, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)			Published 1999

Inventor/Author	Country	Issue/Publication No.	Filed Date
specifications: High-speed Physical Layer in the 5 GHz Band"			
IEEE STANDARD FOR INFORMATION TECHNOLOGY. ANSI/ IEEE "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band ANSI/ IEEE "IEEE Std 802.11b-1999."			Published 1999
IEEE STANDARD FOR INFORMATION TECHNOLOGY. ANSI/ IEEE "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band"			Published 2003
Li et al., "802.11n Channel Model Validation"			Published November 2003

VIII. THE '469 PATENT IS INVALID UNDER 35 U.S.C. §§ 102 AND/OR 103

A. The '469 Patent Priority Date

Plaintiff's Infringement Contentions allege the asserted claims of the '469 Patent have a filing date of September 29, 2003. Southwest disputes the alleged priority date of September 29, 2003. In particular, several aspects of the claims are not adequately supported by the written description of the provisional application. In one example, asserted claim 32 recites "[t]he Internet Hotspot of claim 25, wherein the wireless connection is one of an 802.11a wireless area network, an 802.11b wireless area network, an 802.11g wireless area network, **and an 802.11n wireless area network**." But the 802.11g standard was not adopted until 2009, and indeed had not even

been drafted by September 2003. The patentee could not have been in possession of a system employing a standard that did not, at the time, exist. And the patentee made clear that provisional application No. 60/507,022, on which the claimed priority date is premised discloses only the then-available 802.11b, 802.11g, and 802.11a standards:

> Uses existing technology combined with this unique application. Readily available network technology. Utilizes 802.11b, 802.11g or 802.11a in the present embodiment.

09/29/2003 Specification.

Although it disputes the alleged priority date, Southwest nevertheless applies September 29, 2003, as the priority date for the purpose of these Preliminary Invalidity Contentions, as the Asserted Patent is invalid under 35 U.S.C. Sections 102 and 103 even if it were entitled to an artificially early date of priority. To the extent that Intellectual Ventures modifies its alleged priority date, Southwest may respond and challenge that date to the extent required by law to satisfy its burden.

B. Identification of Prior Art that Anticipates or Renders Obvious the Asserted Claims of the '469 Patent

Southwest contends that the prior art references listed below and charted in Exhibits 469-1 – 469-13 anticipate and/or render obvious, alone or in combination, one or more of the asserted claims of the '469 Patent.

Prior Art Reference	Publication Date
Airpath Wireless Hot Spot	At least by June 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) and (g).
U.S. Patent No. 6,894,990 ("Agarwal")	Filed – October 13, 2000 Issued – May 17, 2005

Prior Art Reference	Publication Date
	This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0138625 ("Bruner")	Filed – March 21, 2001 Published – September 26, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), § 102(b) and § 102(e).
U.S. Patent No. 6,445,777 ("Clark")	Filed – December 21, 1998 Issued – September 3, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (b), (e) and (g).
Connexion by Boeing	At least by June 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (b) and (g).
DeSanctis, et al., "Aeronautical Communications for Personal and Multimedia Services via Satellite," AM. INST. OF AERONAUTICAL. AND ASTRONAUTICS ("DeSanctis")	Published at least by April 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (b), and (g).
Jahn, et al., "Dimensioning of Aeronautical Satellite Services," AM. INST. OF AERONAUTICAL. AND ASTRONAUTICS ("Jahn I")	Published at least by October 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) and (g).
Jahn, et al., "Passenger Multimedia Service Concept via Future Satellite Systems" DGLR JAHRBUCH ("Jahn II")	Published – 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (b) and (g).
U.S. Patent Application No. 2003/0051041 ("Kalavade")	Filed – August 6, 2002 Published – March 13, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (b), (e), and (g).
Maritime Telecommunications Network Cruise Ship Internet Services	At least by June 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) and (g).

Prior Art Reference	Publication Date
U.S. Patent Application No. 2007/0008937 ("Mody")	Filed – July 21, 2004 Priority – July 22, 2003 Published – January 11, 2007 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (e), and (g).
U.S. Patent Application No. 2003/0055975 ("Nelson")	Filed – June 19, 2001 Published – March 28, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (e) and (g).
U.S. Patent Application No. 2004/0109449 ("Seo")	Filed – August 15, 2003 Published – June 10, 2004 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (e), and (g).
Suwanateep, Manasuda, "Inflight Internet Connectivity Technology and Its Application on E-Commerce," U. OF THAILAND (2002) ("Suwanateep")	Published at least by July 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (b), and (g).
Tenzing Global	At least by April 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (b), and (g).
Wireless Cabin	At least by February 9, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) and (g).
U.S. Patent Application Publication No. 2003/0050041 ("Wu")	Filed – September 7, 2001 Published – March 13, 2003 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a), (e), and (g).

Plaintiffs assert claims 24, 25, 26, 28, and 32 of the '469 Patent against Southwest in this lawsuit. These claims are invalid because the '469 Patent fails to meet one or more of the

requirements for patentability. The individual bases for invalidity are provided below and in Exhibits 469-1 – 469-13. Each of the prior art references, the underlying work, and/or the underlying apparatus or method qualifies as prior art under one or more sections of Pre-AIA 35 U.S.C. §§ 102 and/or 103.

In an effort to focus the issues, Southwest has cited representative portions of identified references, even where a reference may contain additional disclosures or support for a particular claim element. Each and every disclosure of the same limitation in the same reference is not necessarily identified. In addition, persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature and in the context of their knowledge as persons of ordinary skill in the art. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other information within the reference, along with other publications and their general scientific knowledge and experience in the field. Thus, each prior art reference identified herein should be understood in the context of the reference itself in conjunction with anything incorporated by reference or described as relevant technology by the reference, any documents that are referred to for further understanding by the reference, any product embodying the reference, and any testimony by those with knowledge of the reference, such as the named authors and inventors. Southwest may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited. Southwest may also rely on uncited portions of the prior art references, other disclosed publications, and the testimony of experts to establish that a person of ordinary skill in the art would have been motivated to modify or combine certain of the cited references so as to render the claims obvious.

Some or all of the asserted claims of the ‘469 Patent are invalid as anticipated under Pre-AIA 35 U.S.C. § 102 in view of certain ones of the prior art references identified above as detailed in the claim charts included in Exhibits 469-1 – 469-13, which identify specific examples of where each limitation of the asserted claims of the ‘469 Patent is found in the prior art references. As explained above, the cited portions of prior art references identified in the attached claim charts are exemplary in nature and representative of the content and teaching of the prior art references, and should be understood in the context of the reference as a whole and as they would be understood by a person of ordinary skill in the art.

To the extent any item of prior art cited above is deemed not to disclose, explicitly or inherently, any limitation of an asserted claim of the ‘469 Patent, Southwest reserves the right to argue that any difference between that prior art and the corresponding patent claim would have been either inherent in the art or obvious to a person of ordinary skill in the art, even if Southwest has not specifically denoted that the art is to be combined with the knowledge of a person of ordinary skill in the art.

To the extent Plaintiffs argue that any limitation of the asserted claims of the ‘469 Patent is not explicitly disclosed by an item of prior art identified above and in Exhibits 469-1 – 469-13, any purported differences are such that the claimed subject matter as a whole would have been obvious to one skilled in the art at the time of the alleged invention, in view of the state of the art and knowledge of those skilled in the art. The item of prior art would, therefore, render the relevant claims invalid as obvious under Pre-AIA 35 U.S.C. § 103.

To the extent any of the foregoing references are found not to anticipate the asserted claims of the ‘469 Patent, the foregoing references render the asserted claims obvious, either alone or in combination with one or more of the other references identified above. One of ordinary skill in the

art would have been motivated to combine the various references cited herein so as to disclose each and every limitation of the asserted claims of the '469 Patent.

The prior art identified in Exhibits 469-1 – 469-13 renders one or more asserted claims of the '469 Patent obvious when the references are read alone or in combination with one or more of each other and/or when read in view of the state of the art and knowledge of those skilled in the art.

To the extent that Plaintiff argues that any asserted claim of the '469 Patent is not disclosed or rendered obvious by any reference charted in Exhibits 469-1 – 469-13, the asserted claims nevertheless would have been obvious in light of one or more other references charted in Exhibits 469-1 – 469-13 and in view of the knowledge, experience, and understanding of persons of ordinary skill in the art.

Each and every reference identified is also relevant to the state of the art at the time of the alleged invention. Any of the references disclosed above may be combined with one another to render obvious (and therefore invalid) each of the asserted claims of the '469 Patent. Southwest may rely upon a subset of the above identified references, or all of the references identified above, including all references charted in Exhibits 469-1 – 469-13, for purposes of obviousness depending on the Court's claim construction, positions taken by Plaintiffs during this litigation, and further investigation and discovery.

1. Example Combinations of the Prior Art

The table below provides an identification of exemplary combinations of prior art references that, based on Southwest's present understanding of Plaintiff's Infringement Contentions and apparent reading of the claims, render obvious one or more of the asserted claims of the '469 Patent. The combinations of references identified below are non-exhaustive, and additional examples are apparent from other combinations of references charted in Exhibits 469-1

– 469-13. Although certain references charted in Exhibits 469-1 – 469-13 are listed variously in the left or right column below, Southwest does not thereby imply any restriction as to which reference is primary and which reference is secondary in such a combination. Southwest reserves the right to rely on either identified reference in the identified combinations as a primary and/or secondary reference.

Exhibit	Reference	Combination Reference(s)
469-1	Agarwal	Airpath Wireless Hot Spot, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-2	Airpath Wireless Hot Spot	Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-3	Bruner	Airpath Wireless Hot Spot, Agarwal, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more other references as charted in Exhibits 469-1 – 469-13.
469-4	Clark	Airpath Wireless Hot Spot, Agarwal, Bruner, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.

Exhibit	Reference	Combination Reference(s)
469-5	Connexion by Boeing	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-6	Jahn I	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-7	Jahn II	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-8	Nelson	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-9	Maritime Telecommunications Network Cruise Ship Internet Services	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-10	Suwanateep	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services,

Exhibit	Reference	Combination Reference(s)
		Nelson, Seo, Tenzing Global, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-11	Tenzing Global	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Wireless Cabin, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-12	Wireless Cabin	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wu, and/or one or more references as charted in Exhibits 469-1 – 469-13.
469-13	Wu	Airpath Wireless Hot Spot, Agarwal, Bruner, Clark, Connexion by Boeing, DeSanctis, Jahn II, Kalavade, Mody, MTN Cruise Ship Internet Services, Nelson, Seo, Suwanateep, Tenzing Global, Wireless Cabin, and/or one or more references as charted in Exhibits 469-1 – 469-13.

It will be understood that, given the similarities between the art, any of Agarwal, Airpath Wireless Hot Spot, Bruner, Clark, Connexion by Boeing, Jahn I, Jahn II, Nelson, Maritime Telecommunications Network Cruise Ship Internet Services, Suwanateep, Tenzing Global, Wireless Cabin, and/or Wu could be combined with one or more of each other, for example based on the disclosures shown in Exhibits 469-1 – 469-13. It would have been obvious to combine these references for reasons that are readily apparent to one of ordinary skill in the art, and for reasons explained in additional detail in the sections that follow.

2. The Claims Recite Well-Known Concepts that Would Have Been Obvious to Implement

As noted above and *supra* II.F the asserted claims of the '469 Patent represent nothing more than an arrangement of well-known concepts and features and add nothing to the state of the art at the time of the alleged invention. For each of the concepts represented by the claims, it would have been inherently obvious to add to the teachings of any number of known references and products, including to any of the primary references charted in Exhibits 469-1 – 469-13 to the extent that such reference is alleged to be missing such concept or feature.

3. Additional Reasons to Combine

In addition to the information contained elsewhere in these invalidity contentions, including in Exhibits 469-1 – 469-13, Southwest has identified additional motivations and reasons to combine the various references. Southwest notes that in *KSR Int'l Co. v. Teleflex Inc.*, the Supreme Court held that "[i]n determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." 550 U.S. 398, 419 (2007).

KSR further illustrated several ways in which the subject matter of a patent claim may be shown to be obvious. For instance, "[o]ne of the ways in which a patent's subject matter can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *KSR*, 127 S.Ct. 1727, 1742. The Supreme Court held that it was error to "look only to the problem the patentee was trying to solve." *Id*. Rather, "[t]he question is not whether the combination was obvious to the patentee but whether the combination was obvious to a person with ordinary skill in the art. Under the correct analysis, any need or problem known in the field of endeavor at the time of invention and addressed

by the patent can provide a reason for combining the elements in the manner claimed." *Id*. Further, "[c]ommon sense teaches … that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* Further, a showing that a combination of elements was "obvious to try" may show that it was obvious under § 103. *Id.* For instance, the Supreme Court held that "predictable solutions" using a combination of "known options" may render the subject matter of a patent claim obvious.

When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product is not of innovation but of ordinary skill and common sense. *Id.* The ways in which the subject matter of a patent claim may be shown to be obvious, identified by *KSR*, are merely illustrative. The main thrust of *KSR* was that "[r]igid preventative rules that deny factfinders recourse to common sense"—such as the overturned "teaching, suggestion, or motivation" test—are neither necessary under our case law nor consistent with it." *KSR*, 127 S.Ct. 1727, 1742-43; citing with approval, *e.g.*, *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F. 3d 1356, 1367 (2006) ("Our suggestion test is in actuality quite flexible and not only permits, but requires, consideration of common knowledge and common sense"); *Alza Corp. v. Mylan Labs., Inc.*, 464 F. 3d 1286, 1291 (2006) ("There is flexibility in our obviousness jurisprudence because a motivation may be found implicitly in the prior art. We do not have a rigid test that requires an actual teaching to combine...").

Although Southwest notes that there is no longer a rigid requirement regarding motivation to combine under *KSR*, Southwest provides the following statements regarding motivation to

combine. Multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 469-1 – 469-13 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one.

The combination of prior art references identified in these contentions would have been obvious because the combination represents the known potential options with a reasonable expectation of success. Additionally, one of ordinary skill in the art would have been motivated to create combinations identified in these contentions using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or teaching, suggestion, or motivation in the prior art generally. Also, market forces in the industry, and the desire to improve features and performance, would motivate the addition of features to products as they become available, become less expensive, become more commonly used, provide better performance, reduce costs, or predictably achieve other clearly desirable results.

Moreover, in addition to the information contained elsewhere in these contentions, and, in particular, charted in Exhibits 469-1 – 469-13, Southwest has identified additional motivations and reasons to combine the various references cited herein. In particular, multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of

the references charted in Exhibits 469-1 – 469-13 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one. The combination of prior art references identified in these contentions would have been obvious because the claimed combinations represent the known potential options with a reasonable expectation of success.

To the extent that Plaintiffs contends that any of the above-identified references does not expressly or inherently disclose a particular element or limitation, it would have been obvious to a person of ordinary skill in the art to modify the references or to combine their teachings with other prior art references, including the prior art references in these invalidity contentions, in a manner that renders such claims obvious and invalid.

For example, many of the prior art references charted in Exhibits 469-1 – 469-13 share a goal of providing Internet users with airborne, seaborne, or otherwise remote access to the Internet by way of their own PDAs or laptops. Various references identified in these Preliminary Infringement Contentions disclose, well before the alleged critical date of the '469 patent, installation of paid or otherwise access-limited Internet hotspots in remote locations for users who desire access "[w]hen away from home or office." Wu at [0003]; *id.* at [0004] ("Over recent years, the field of wireless communications has enjoyed tremendous growth.").

For example, Wu discloses "a network system capable of providing prepaid wireless remote access service" which "obtains customer-inputted identification information related to the prepaid account" using an "authentication client." Wu at [0001], [0009], [0033] (noting that "virtually any type of prepaid arrangement or prepayment system may be effected" including, in an alternative embodiment, that funds be "deducted immediately and electronically from a credit card, a debit card, a check card, a charge account, or other forms of remote payment systems." Wu further teaches that subscriber access could be easily handled by "a commercially available device such as the ISS-2000 Internet Subscriber Server manufactured by Handlink Technology, Inc." Wu at [0043].

As another example, Bruner discloses "an airline-based system for providing electronic commerce to passengers during flights" and that "it will be understood by those skilled [sic] the art that the invention is generally applicable to any remote vehicle or building including aircraft, ships, trains and hotels." Bruner at [0001]. According to Bruner, "airborne users 13 are connected to the system via the aircraft based user terminals which are connected to an airline-based remote server computer" by way of an Airborne Data Server (ADS).

As another example, Kalavade teaches the use of commercially-available client access servers "at the hotspot, such as, e.g., a Cisco Router or a Nokia Access controller intercepts the session and sends a pre-defined web page to the user. This typically queries the user for a login and a password . . . Once authenticated, the RADIUS server responds with an Access Response to the RADIUS client." Kalavade at [0056], [0207].

As another example, Jahn I and Jahn II disclose a system for "travellers in aircraft on the move" including "wireless access segments in the aircraft cabin, namely a wireless LAN according to IEEE 802.11b standard," a "satellite segment for interconnection of the cabin with the terrestrial

telecom networks," and on-board service integration. Jahn II at 2-3. Jahn explains that, "since IP traffic at IP connection level can be monitored only before the IS stream multiplexing, it is envisaged that AAA functions must be located in the cabin to monitor for instance, volume-based accounting." *Id*. at 8.

As set forth above and *supra* II.F, the market and industry for Internet hotspots was well established and known by one of ordinary skill in the art. A person of ordinary skill would know, for example, to consider any feature of remote Internet hotspot designs already implemented by major industry players, including systems such as Airpath Hot Spot, Connexion by Boeing, or MTN for Cruise Ships. A person of ordinary skill would know that installation of an Internet hotspot could be achieved using "out of the box" hardware and software, and could be then improved upon or customized using any number of known techniques to achieve specific business objectives such as accepting credit cards or offering prepaid or subscription access to a commodity service.

* * *

In addition to the specific combinations of prior art and the specific combinations of groups of prior art disclosed above, Southwest reserves the right to rely on any other combination of any prior art references disclosed herein. Southwest further reserves the right to rely upon combinations of prior art references disclosed within the prosecution history of the references cited herein.

The references charted in Exhibits 469-1 – 469-13 disclose the elements of the asserted claims of the '469 Patent explicitly and/or inherently and may also be relied upon to show the state of the art at the time of the alleged invention. The suggested obviousness combinations are

provided in the alternative to Southwest's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not itself anticipatory.

C. OGP: Charts Setting Forth Where in the Prior Art References Each Element of the Asserted Claims Are Found

Pursuant to the Court's OGP, charts setting forth where specifically in each alleged item of prior art each element of each Asserted Claim of the '469 Patent is found are attached as Exhibits 469-1 – 469-13.

D. Other Relevant Prior Art References

In addition to the references identified above and charted in Exhibits 469-1 – 469-13, each of the references below is relevant to the '469 Patent, as it relates to the state of the art. Southwest reserves the right to amend its contentions to rely on any of the below references alone or in combination with any other references identified in these contentions.

Inventor/Author	Country	Issue/Publication No.	Filed Date
B. Reilly Barry et al.	US	6,615,258	September 24, 1998
Burt Liebowitz et al.	US	5,757,784	January 4, 1996
David D. Otten et al.	US	7,174,127	September 17, 2002
Jeffrey A. Frisco et al.	US	6,208,307	April 7, 2000
Jeyhan Karaoguz et al.	US	6,885,859	December 9, 2002
John Howard et al.	US	2003/0204610	April 30, 2003
John Thomasson et al.	US	2001/0023435	December 20, 2000
Kazuhiro Hara et al.	US	6,560,221	March 6, 1998
Khalid Monir A. El-Rafie et al.	US	6,968,394	March 21, 2000
Michael Massey	US	2001/0026537	February 23, 2001
Nikhil M. Deshpande et al.	US	2002/0176579	May 24, 2001
Patrick Meenan et al.	US	7,283,505	October 31, 2002
Per Wahlberg	US	2004/0137841	April 19, 2002
Robert Davis et al.	US	2002/0154055	April 15, 2002
Selim Rakib	US	2002/0031120	July 3, 2001
Shailendra Depura et al.	US	2003/0084051	July 26, 2002
Yi Gong et al.	US	2005/0032531	August 6, 2003
Conforto et al., "Suited/GMBS System: Architecture and Mobile Terminal"			January 2001

Inventor/Author	Country	Issue/Publication No.	Filed Date
Holzbock et al., "Land-Mobile Demonstration of Multimedia IP Services in Multi-Segment Access Networks," AM. INST. OF AERONAUTICAL. AND ASTRONAUTICS			At least by May 12, 2002
Holzbock et al., "Evolution of Aeronautical Communications for Personal and Multimedia Services" IEEE COMMS. MAGAZINE			July 2003
Jovanovic et al., "Joint Tactical Radio - Aspects of Standardization," SCIENTIFIC-TECHNICAL REVIEW, vol. III, no. 2			At least by July 2003
Martin, "Communication Satellites (1958-1992)," THE AEROSPACE CORPORATION			At least by December 31, 1991

IX. THE '582 PATENT IS INVALID UNDER 35 U.S.C. §§ 102 AND/OR 103

A. The '582 Patent Priority Date

Plaintiff's Infringement Contentions allege the asserted claims of the '582 Patent have a filing date of March 13, 2002.

B. Identification of Prior Art that Anticipates or Renders Obvious the Asserted Claims of the '582 Patent

Southwest contends that the prior art references listed below and charted in Exhibits 582-1 – 582-6 anticipate and/or render obvious, alone or in combination, one or more of the asserted claims of the '582 Patent.

Prior Art Reference	Publication Date
U.S. Patent No. 6,304,866 ("Chow")	Filed – June 27, 1997 Issued – October 16, 2001

Prior Art Reference	Publication Date
	This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0019844 ("Kurowski")	Filed – January 12, 2001 Published – February 14, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 6,330,583 ("Reiffin")	Filed – September 9, 1994 Issued – December 11, 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 6,212,617 ("Hardwick")	Filed – June 30, 1998 Issued – April 3, 2001 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0103889 ("Markson")	Filed – June 19, 2001 Published – August 1, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent Application Publication No. 2002/0091786 ("Yamaguchi")	Filed – November 1, 2001 Published – July 11, 2002 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(e).
U.S. Patent No. 5,765,146 ("Wolf")	Filed – November 4, 1993 Issued – June 9, 1998 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b) and § 102(e).
EP Patent Application Publication No. EP0377993 ("Lorie")	Published – July 18, 1990 This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b).
P. Maheshwari, "Improving task scheduling for larger grain execution of parallel functional	Published January 8, 1993

Prior Art Reference	Publication Date
programs," [1993] Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences, Wailea, HI, USA, 1993, pp. 594-602 vol.2, doi: 10.1109/HICSS.1993.284066. ("Maheshwari")	This reference is prior art under at least Pre-AIA 35 U.S.C. § 102(b). Retrieved from https://ieeexplore.ieee.org/document/284066
Gamma System - University of Wisconsin-Madison	At least as early as 1990 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) or (g).
Volcano Parallel Database System - University of Wisconsin-Madison	At least as early as 1990 This reference is prior art under at least Pre-AIA 35 U.S.C. §§ 102(a) or (g).

Plaintiffs assert claims 1, 2, 5, and 7 of the '582 Patent against Southwest in this lawsuit. These claims are invalid because the '582 Patent fails to meet one or more of the requirements for patentability. The individual bases for invalidity are provided below and in Exhibits 582-1 – 582-6. Each of the prior art references, the underlying work, and/or the underlying apparatus or method qualifies as prior art under one or more sections of Pre-AIA 35 U.S.C. §§ 102 and/or 103.

In an effort to focus the issues, Southwest has cited representative portions of identified references, even where a reference may contain additional disclosures or support for a particular claim element. Each and every disclosure of the same limitation in the same reference is not necessarily identified. In addition, persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature and in the context of their knowledge as persons of ordinary skill in the art. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other information within the reference, along with other publications and their general scientific

knowledge and experience in the field. Thus, each prior art reference identified herein should be understood in the context of the reference itself in conjunction with anything incorporated by reference or described as relevant technology by the reference, any documents that are referred to for further understanding by the reference, any product embodying the reference, and any testimony by those with knowledge of the reference, such as the named authors and inventors. Southwest may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited. Southwest may also rely on uncited portions of the prior art references, other disclosed publications, and the testimony of experts to establish that a person of ordinary skill in the art would have been motivated to modify or combine certain of the cited references so as to render the claims obvious.

Some or all of the asserted claims of the '582 Patent are invalid as anticipated under Pre-AIA 35 U.S.C. § 102 in view of certain ones of the prior art references identified above as detailed in the claim charts included in Exhibits 582-1 – 582-6, which identify specific examples of where each limitation of the asserted claims of the '582 Patent is found in the prior art references. As explained above, the cited portions of prior art references identified in the attached claim charts are exemplary in nature and representative of the content and teaching of the prior art references, and should be understood in the context of the reference as a whole and as they would be understood by a person of ordinary skill in the art.

To the extent any item of prior art cited above is deemed not to disclose, explicitly or inherently, any limitation of an asserted claim of the '582 Patent, Southwest reserves the right to argue that any difference between that prior art and the corresponding patent claim would have been either inherent in the art or obvious to a person of ordinary skill in the art, even if Southwest

has not specifically denoted that the art is to be combined with the knowledge of a person of ordinary skill in the art.

To the extent Plaintiffs argue that any limitation of the asserted claims of the '582 Patent is not explicitly disclosed by an item of prior art identified above and in Exhibits 582-1 – 582-6, any purported differences are such that the claimed subject matter as a whole would have been obvious to one skilled in the art at the time of the alleged invention, in view of the state of the art and knowledge of those skilled in the art. The item of prior art would, therefore, render the relevant claims invalid as obvious under Pre-AIA 35 U.S.C. § 103.

To the extent any of the foregoing references are found not to anticipate the asserted claims of the '582 Patent, the foregoing references render the asserted claims obvious, either alone or in combination with one or more of the other references identified above. One of ordinary skill in the art would have been motivated to combine the various references cited herein so as to disclose each and every limitation of the asserted claims of the '582 Patent.

The prior art identified in Exhibits 582-1 – 582-6 renders one or more asserted claims of the '582 Patent obvious when the references are read alone or in combination with one or more of each other and/or when read in view of the state of the art and knowledge of those skilled in the art.

To the extent that Plaintiff argues that any asserted claim of the '582 Patent is not disclosed or rendered obvious by any reference charted in Exhibits 582-1 – 582-6, the asserted claims nevertheless would have been obvious in light of one or more other references charted in Exhibits 582-1 – 582-6 and in view of the knowledge, experience, and understanding of persons of ordinary skill in the art.

Each and every reference identified is also relevant to the state of the art at the time of the alleged invention. Any of the references disclosed above may be combined with one another to render obvious (and therefore invalid) each of the asserted claims of the '582 Patent. Southwest may rely upon a subset of the above identified references, or all of the references identified above, including all references in charted in Exhibits 582-1 – 582-6, for purposes of obviousness depending on the Court's claim construction, positions taken by Plaintiffs during this litigation, and further investigation and discovery.

1. Example Combinations of the Prior Art

The table below provides an identification of exemplary combinations of prior art references that, based on Southwest's present understanding of Plaintiff's Infringement Contentions and apparent reading of the claims, render obvious one or more of the asserted claims of the '582 Patent. The combinations of references identified below are non-exhaustive, and additional examples are apparent from other combinations of references charted in Exhibits 582-1 – 582-6. Although certain references charted in Exhibits 582-1 – 582-6 are listed variously in the left or right column below, Southwest does not thereby imply any restriction as to which reference is primary and which reference is secondary in such a combination. Southwest reserves the right to rely on either identified reference in the identified combinations as a primary and/or secondary reference.

Exhibit	Reference	Combination Reference(s)
582-1	Chow	Reiffin, Kurowski, Yamaguchi, Wolf, Lorie, and/or Markson, and/or one or more references as charted in Exhibits 582-1 – 582-6
582-2	Reiffin	Chow, Kurowski, Yamaguchi, Wolf, Lorie, and/or Markson, and/or one or

Exhibit	Reference	Combination Reference(s)
		more references as charted in Exhibits 582-1 – 582-6
582-3	Hardwick	Reiffin, Kurowski, Yamaguchi, Wolf, Lorie, Chow, and/or Markson, and/or one or more references as charted in Exhibits 582-1 – 582-6
582-4	Maheshwari	Reiffin, Kurowski, Yamaguchi, Wolf, Lorie, Chow, and/or Markson, and/or one or more references as charted in Exhibits 582-1 – 582-6
582-5	Gamma System	Reiffin, Kurowski, Yamaguchi, Wolf, Lorie, Chow, and/or Markson, and/or one or more references as charted in Exhibits 582-1 – 582-6
582-6	Volcano Parallel Database System	Reiffin, Kurowski, Yamaguchi, Wolf, Lorie, Chow, and/or Markson, and/or one or more references as charted in Exhibits 582-1 – 582-6

It will be understood that, given the similarities between the art, any of Chow, Reiffin, Hardwick, Kurowski, Yamaguchi, Wolf, Lorie, Markson, Maheshwari, the Gamma System, and/or the Volcano Parallel Database System could be combined with one or more of each other, for example based on the disclosures shown in Exhibits 582-1 – 582-6. It would have been obvious to combine these references for reasons that are readily apparent to one of ordinary skill in the art, and for reasons explained in additional detail in the sections that follow.

2. The Claims Recite Well-Known Concepts that Would Have Been Obvious to Implement

As noted above, the asserted claims of the ‘582 Patent represent nothing more than an arrangement of well-known concepts and features and add nothing to the state of the art at the time of the alleged invention. For each of the concepts represented by the claims, it would have been inherently obvious to add to the teachings of any number of known references and products,

including to any of the primary references charted in Exhibits 582-1 – 582-6 to the extent that such reference is alleged to be missing such concept or feature.

3. Additional Reasons to Combine

In addition to the information contained elsewhere in these invalidity contentions, including in Exhibits 582-1 – 582-6, Southwest has identified additional motivations and reasons to combine the various references. Southwest notes that in *KSR Int'l Co. v. Teleflex Inc.*, the Supreme Court held that "[i]n determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." 550 U.S. 398, 127 S.Ct. 1727, 1742 (2007).

KSR further illustrated several ways in which the subject matter of a patent claim may be shown to be obvious. For instance, "[o]ne of the ways in which a patent's subject matter can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *KSR*, 127 S.Ct. 1727, 1742. The Supreme Court held that it was error to "look only to the problem the patentee was trying to solve." *Id*. Rather, "[t]he question is not whether the combination was obvious to the patentee but whether the combination was obvious to a person with ordinary skill in the art. Under the correct analysis, any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id*. Further, "[c]ommon sense teaches … that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* Further, a showing that a combination of elements was "obvious to try" may show that it was obvious under § 103. *Id.* For instance, the Supreme

Court held that "predictable solutions" using a combination of "known options" may render the subject matter of a patent claim obvious.

When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. *Id.* The ways in which the subject matter of a patent claim may be shown to be obvious, identified by *KSR*, are merely illustrative. The main thrust of *KSR* was that "[r]igid preventative rules that deny factfinders recourse to common sense"—such as the overturned "teaching, suggestion, or motivation" test—are neither necessary under our case law nor consistent with it." *KSR*, 127 S.Ct. 1727, 1742-43; citing with approval, e.g., *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F. 3d 1356, 1367 (2006) ("Our suggestion test is in actuality quite flexible and not only permits, but requires, consideration of common knowledge and common sense"); *Alza Corp. v. Mylan Labs., Inc.*, 464 F. 3d 1286, 1291 (2006) ("There is flexibility in our obviousness jurisprudence because a motivation may be found implicitly in the prior art. We do not have a rigid test that requires an actual teaching to combine...").

Although Southwest notes that there is no longer a rigid requirement regarding motivation to combine under *KSR*, Southwest provides the following statements regarding motivation to combine. Multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 582-1 – 582-6 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one

of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one.

The combination of prior art references identified in these contentions would have been obvious because the combination represents the known potential options with a reasonable expectation of success. Additionally, one of ordinary skill in the art would have been motivated to create combinations identified in these contentions using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or teaching, suggestion, or motivation in the prior art generally. Also, market forces in the industry, and the desire to improve features and performance, would motivate the addition of features to products as they become available, become less expensive, become more commonly used, provide better performance, reduce costs, or predictably achieve other clearly desirable results.

Moreover, in addition to the information contained elsewhere in these contentions, and, in particular, charted in Exhibits 582-1 – 582-6, Southwest has identified additional motivations and reasons to combine the various references cited herein. In particular, multiple teachings, suggestions, and/or reasons to modify any of the references and/or to combine any two or more of the references charted in Exhibits 582-1 – 582-6 come from many sources, including the prior art (specific and as a whole), common knowledge, common sense, predictability, expectations, industry trends, design incentives or need, market demand or pressure, market forces, obvious to try, the nature of the problem faced, and/or knowledge possessed by one of ordinary skill. In addition, it would have been obvious to try combining the prior art references identified above

because there were only a finite number of predictable solutions and/or because known work in one field or endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one. The combination of prior art references identified in these contentions would have been obvious because the claimed combinations represent the known potential options with a reasonable expectation of success.

To the extent that Plaintiffs contend that any of the above-identified references does not expressly or inherently disclose a particular element or limitation, it would have been obvious to a person of ordinary skill in the art to modify the references or to combine their teachings with other prior art references, including the prior art references in these invalidity contentions, in a manner that renders such claims obvious and invalid.

In particular, various references identified above disclose, well before the critical date of the '582 Patent, distributed computing systems that already solved the '582 Patent's goal of decomposing linear computational processes that currently use a single computer into equivalent parallel processes that can use any number of heterogeneous computers.

For example, Chow discloses distributed multi-processing systems and database operations. Chow at 2:42-46. Chow's disclosure provides a multiprocessing system "to perform an aggregate task as quickly as possible, with minimum workload skew among independently operating processing elements." *See* Chow at 2:1-7.

As another example, Reiffin discloses distributed multi-processing systems wherein a large compute-intensive task may be partitioned into subtasks for parallel execution. Reiffin at 1:7-14. Reiffin provides that "a large compute-intensive task may be partitioned into subtasks which are then distributed among a plurality of personal computer or workstations for parallel execution of the subtasks." *See* Reiffin at 1:9-12.

As another example, Kurowski discloses large-scale network-distributed computing systems. Kurowski at [0003]. Kurowski's disclosure provides "[a] distributed computing system achieves a highly distributed environment where very large computation intensive tasks are broken down into thousands of sub-tasks and then distributed to thousands of clients running on a variety of computers across the Internet." *See* Kurowski at Abstract.

As yet another example, Hardwick discloses computer-executable processes following a divide-and-conquer algorithm where the input file is derived from the storage unit and the input is then logically distributed or shared into partitions. The partitions are then distributed to the subtask processors for concurrent processing. *See* Hardwick at Abstract, 10:35-44, claim 1.

As yet another example, Maheshwari discloses a parallel processing system that uses concurrent task partitions, and data is processed in a first-in-first-out (FIFO) manner. *See* Maheshwari at 2-3.

As yet another example, the Gamma System was a parallel relational database system developed at the University of Wisconsin-Madison in the late 1980s. It embodied many concepts that later influenced systems like MapReduce, particularly in how it handled partitioned parallelism for query execution. *See, e.g.*, https://apps.dtic.mil/sti/pdfs/ADA223436.pdf. The Gamma System subdivided data for parallel processing. *See* https://minds.wisconsin.edu/bitstream/handle/1793/11036/file_1.pdf, https://www.cs.ubc.ca/~rap/teaching/504/2022W2/slides/parallel.pdf, https://minds.wisconsin.edu/bitstream/handle/1793/58708/TR635.pdf?sequence=1.

As yet another example, the Volcano Parallel Database System was a parallel database system designed in the late 1980s and early 1990s at the University of Wisconsin–Madison. It served as a foundation for many later research projects and commercial systems. Volcano is best

known not just as a parallel database, but also as an extensible and highly modular query execution engine. Volcano distributes tasks to partitions each with a subtask processor and processes subtasks in parallel. *See* https://paperhub.s3.amazonaws.com/dace52a42c07f7f8348b08dc2b186061.pdf, https://pages.cs.wisc.edu/~shivaram/cs744-readings/volcano.pdf, https://cs-people.bu.edu/mathan/reading-groups/papers-classics/volcano.pdf.

Various references identified above also disclose, well before the critical date of the '582 Patent, file allocation and partitioning, load balancing, using additional processors to carry out file allocation/partitioning and associated distribution, providing an accumulation of output records in an arbitrary order, and input files residing on network attached storage.

For example, Yamaguchi discloses a system in which a program is executed on each of several logical partitions, the load on each of the logical partitions is monitored, and when a logical partition is overloaded, data being distributed by the logical partition is automatically switched for distribution by another logical partition. *See* Yamaguchi at Abstract.

As another example, Wolf discloses using additional processors for performing distributed processing steps. *See* Wolf at 4:63-5:7.

As yet another example, Lorie discloses performing processing in random or arbitrary orders. *See* Lorie at 6:50-56.

As yet another example, Markson discloses a system which logically allocates storage and to the processors and also logically associates logical units from the storage units to those processors, and providing input files on network attached storage. *See* Markson at [0118].

A person of ordinary skill in the art would have been motivated to combine the above references to incorporate the different beneficial technical features provided by each of the above

references. Modifying the above references would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results.

* * *

In addition to the specific combinations of prior art and the specific combinations of groups of prior art disclosed above, Southwest reserves the right to rely on any other combination of any prior art references disclosed herein. Southwest further reserves the right to rely upon combinations of prior art references disclosed within the prosecution history of the references cited herein.

The references charted in Exhibits 582-1 – 582-6 disclose the elements of the asserted claims of the '582 Patent explicitly and/or inherently and may also be relied upon to show the state of the art at the time of the alleged invention. The suggested obviousness combinations are provided in the alternative to Southwest's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not itself anticipatory.

C. OGP: Charts Setting Forth Where in the Prior Art References Each Element of the Asserted Claims Are Found

Pursuant to the Court's OGP, charts setting forth where specifically in each alleged item of prior art each element of each Asserted Claim of the '582 Patent is found are attached as Exhibits 582-1 – 582-6.

D. Other Relevant Prior Art References

In addition to the references identified above and charted in Exhibits 582-1 – 582-6, each of the references below is relevant to the '582 Patent, as it relates to the state of the art. Southwest reserves the right to amend its contentions to rely on any of the below references alone or in combination with any other references identified in these contentions.

Inventor/Author	Country	Issue/Publication No.	Filed Date
Tuneo Iida et al.	US	6,334,137	August 24, 1999
Chad Grey Thompson	US	2002/0099716	January 25, 2001
Yuichi Aiba	JP	JP20003/30959	November 30, 2000
Dianne L. Steiger et al.	US	US2002/040381	September 28, 2001
Russ W. Herrell et al.	US	5,325,493	March 30, 1993
Christopher L. Reeve	US	5,535,393	June 5, 1995
Roberta L. Gotfried	US	5,765,166	June 9, 1998
S. Kuo, M. Winslett, Y. Chen, Y. Cho, M. Subramaniam and K. Seamons, "Parallel input/output with heterogeneous disks," Proceedings. Ninth International Conference on Scientific and Statistical Database Management (Cat. No.97TB100150), Olympia, WA, USA, 1997, pp. 79-90, doi: 10.1109/SSDM.1997.621154.			Pub. Aug. 11-13, 1997 Retrieved from https://ieeexplore.ieee.org/document/621154
David Morrison et al.	US	5,945,990	November 19, 1996
Shawn B. Smith et al.	WO	03/012696	July 29, 2002
James Mathew Rehg et al.	US	6,480,876	May 28, 1998
"Some Thoughts on Parallel Processing," Lynn D. Yarbough, Communications of the ACM, Vol. 3, Issue 10, page 539, October 1, 1960 (available at: https://dl.acm.org/doi/10.1145/367415.367426).			Oct. 1, 1960
"A Survey of Some Theoretical Aspects of Multiprocessing," J. L. Baer, ACM Computing Surveys (CSUR), Volume 5, Issue 1 Pages 31 – 80, March 1, 1973 (available at: https://dl.acm.org/doi/10.1145/356612.356615)			March 1, 1973
"Parallel Sorting Algorithms for Tightly Coupled Multiprocessors," Michael J.			1988

Inventor/Author	Country	Issue/Publication No.	Filed Date
Quinn, Parallel Computing 6 (1988) 349-357.			
"Removing Skew Effect in Join Operation on Parallel Processors," Ron-Chung Hu, Richard R. Muntz, Computer Science Technical Report, University of California, Los Angeles, June 1989.			June 1989
"Parallel Architectures for Database Systems," A.R. Hurson, L.L. Miller, S.H. Pakzad, M.H. Eich, B. Shirazi, Advances in Computers, Vol. 28, pages 107-151, 1989.			1989
"Parallel Algorithms for Merging and Sorting," Narsingh Deo, Dilip SarKar, Information Sciences, Vol. 56, pages 151-161, 1991.			1991
"Parallel Algorithms for lcast Median of Squares Regression," Chong-Wei Xu and Wei-Kei Shiue, Computational Statistics & Data Analysis, Vol. 16, pages 349-362, 1993.			1993
"On the Average Running Time of Odd-Even Merge Sort," Christine Rub, Max Plank-Institute Fur Informtik, Research Report, April 1995.			April 1995
"Parallel Algorithms," Guy E. Blelloch and Bruce M. Maggs, a chapter in "The Computer Science and Engineering Handbook." CRC Press, 1997, ISBN: 0-8493-2909-4. Available at https://dl.acm.org/doi/pdf/10.5555/1882723.1882748.			1997
"An Approach to Parallelizing Isotonic Regression," Anthony J. Kearsley, Richard A. Tapia, Michael W.Trosset, Applied			1996

Inventor/Author	Country	Issue/Publication No.	Filed Date
Mathematics and Parallel Computing, H. Fisher et. al. (eds.), Physica-Verlag Heidelberg (1996), pages 141-147.			
"Part 1: The Parallel Computing Environment, Alice E. Koniges, Morris A. Jette, and David C. Eder, in "Industrial Strength Parallel Computing," Morgan Kaufmann, (2000). ISBN 9781558605404. Available at: http://wayback.cecm.sfu.ca/PSG/book/intro.html and at https://books.google.com/books?id=mWalawBciCQC&pg=PA1&source=gbs_toc_r&cad=2#v=onepage&q&f=false.			2000
GPFS: A Parallel File System, Jason Barkes, Marcelo R. Barrios, Francis Cougard, Paul G. Crumley, Didac Marin, Hari Reddy, Theerapong Thitayanun, IBM International Technical Support Organization, April 1998.			April 1998
"Analysis of First-Come-First-Serve Parallel Job Scheduling," Uwe Schwiegelshohn and Ramin Yahyapour. Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (1998). Available at http://dx.doi.org/10.1145/314613.315031.			1998
Bricker, et al., "Condor Technical Summary," University of Wisconsin – Madison Computer Sciences Department			Pub. Jan. 1992
W. Gentzsch, "Sun Grid Engine: Towards Creating a Compute Power Grid," Sun Microsystems, Inc.			2001

Inventor/Author	Country	Issue/Publication No.	Filed Date
O. Sievert, "A Gentle Introduction to Globus," Prepared for Parallel Computation, CSE160/CSE260, Spring 1999			1999
Anderson, et al., "SETI@home Internet Distributed Computing for SETI," ASP Conference Series, Vol. 213			2000
Bamford, et al., "Architecture of Oracle Parallel Server," Proceedings of the 24th Very Large Database (VLDB) Conference New York, USA, 1998			Pub. 1998
J. Howard, "An Overview of the Andrew File System," Carnegie Mellon University			1988
S. Hotovy, "Workload Evolution on the Cornell Theory Center IBM SP2" Cornell Theory Center			1996
Foster et al., "Wide-Area Implementation of the Message Passing Interface," Parallel Computing 24(12),1735-1749			1998
Heymann et al., "Adaptive Scheduling for Master-Worker Applications on the Computational Grid," Department of Computer Sciences University of Wisconsin – Madison			2000
D. Gelernter and N. Carriero, "How to Write Parallel Programs: A Guide to the Perplexed," *ACM Computing Surverys*, Vol. 21, No. 3			Sept. 1989

X. GROUNDS OF INVALIDITY UNDER 35 U.S.C. § 112

Defendants set forth indefiniteness contentions below, but reserve the right to revise, amend, and/or supplement these indefiniteness contentions as part of the claim construction

briefing process as provided for in the Court's scheduling order. The following is provided in a good faith effort to comply with the Court's scheduling order. These contentions shall not be deemed admissions, whether express or implied, regarding the scope of any claims, or the proper constructions of those claims or any terms recited in those claims. Nor do Defendants waive other defenses pursuant to 35 U.S.C. § 112.

A. U.S. Patent No. 8,332,844

Claims 7 and 11 of the 844 Patent (herein, the "Asserted '844 Claims") are invalid under 35 U.S.C. §§ 112(a) and 112(b), respectively, for at least the reasons discussed below.

The Asserted '844 Claims contain subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the applicants, at the time the application was filed, had possession of the claimed invention. In addition, the Asserted '844 Claims contain subject matter that was not described in the specification in such a way as to enable one of ordinary skill in the art to make and/or use the invention. Further, the Asserted '844 Claims fail to particularly point out and distinctly claim the invention, such that the claim's scope cannot be ascertained with reasonable certainty.

To the extent that the Asserted 844 Claims are construed broadly to cover undisclosed embodiments, the claims are invalid because the '844 Patent specification does not enable a person skilled in the art to practice the full scope of the purported invention, nor does the 844 Patent specification show the applicants had within their possession the full scope of the Asserted '844 Claims. Further, to the extent that any of the Asserted '844 Claims are construed sufficiently broadly to cover one or more of the various accused products identified in IV's infringement contentions, such claims are invalid pursuant to at least 35 U.S.C. § 112(a).

Southwest contends that the following terms of asserted claims 7 and 11 are indefinite under 35 U.S.C. § 112(b), and/or lack written description support under 35 U.S.C. § 112(a):

1. **“only additional data blocks not previously contained in said root image” (Claim 7)**

Indefiniteness: This phrase requires that the leaf image contains “only additional” data blocks that were not previously in the root image. However, the patent does not provide a clear definition of what constitutes an “additional” block versus a modified block. If a block is modified rather than newly created, does it still count as “additional”? The claim does not specify how to determine whether a block was previously contained in the root image. For example: If a block has the same content but is stored in a different location, is it still “previously contained”? If a block is partially altered, does it become “additional,” or is it still part of the original root image? Without objective boundaries, a POSA is unable to determine whether a system meets this limitation.

Written Description: The specification describes that leaf images store additional data blocks and modifications to the root image, but does not explicitly define “only additional.” The term “only additional” appears to exclude modifications, yet the patent acknowledges that leaf images may store modifications.

1. **“accessed by at least one of said compute nodes” (Claim 7)**

Indefiniteness: This phrase refers to caching blocks that have been “accessed” by at least one compute node, but the term “accessed” is undefined and ambiguous. What type of access qualifies? Read operations only? Write operations? Execution of code within a block? Any attempted access, even if the operation is unsuccessful? The claim does not clarify whether a single access event is sufficient to trigger caching, or if caching occurs only after repeated access.

2. "merging the blocks of said root image with the blocks of respective leaf images" (Claim 11)

Indefiniteness: The claim does not specify what "merging" means in this context. Overwriting blocks in the root image with leaf image data? Appending new blocks without modifying existing ones? Interleaving blocks from the root and leaf images?

3. "to create cohesive respective application environments" (Claim 11)

Indefiniteness: What makes an application environment "cohesive"? Data consistency across all compute nodes? Performance efficiency? Uniform file structure? Without an objective test for determining whether an environment is "cohesive", a POSITA cannot reasonably ascertain the scope of this claim term.

Written Description: The specification never explains what a "cohesive" environment entails.

4. "at an operational level between file systems" (Claim 11)

Indefiniteness: What is meant by "operational level"? Kernel-level operations? Middleware interactions? File system APIs? Without a defined meaning, the phrase is open to multiple interpretations, making the claim scope uncertain.

Written Description: The specification never uses the phrase "operational level," and in the context of the claim, the phrase lacks written description support.

B. U.S. Patent No. 8,407,722

Claims 14, 16, 17 and 18 of the '722 Patent (herein, the "Asserted '722 Claims") are invalid under 35 U.S.C. §§ 112(a) and 112(b), respectively, for at least the reasons discussed below.

The Asserted '722 Claims contain subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the applicants, at the time the application was filed, had possession of the claimed invention. In addition, the Asserted

‘722 Claims contain subject matter that was not described in the specification in such a way as to enable one of ordinary skill in the art to make and/or use the invention. Further, the Asserted ‘722 Claims fail to particularly point out and distinctly claim the invention, such that the claim’s scope cannot be ascertained with reasonable certainty.

To the extent that the Asserted ‘722 Claims are construed broadly to cover undisclosed embodiments the claims are invalid because the ‘722 Patent specification does not enable a person skilled in the art to practice the full scope of the purported invention, nor does the ‘722 Patent specification show the applicants had within their possession the full scope of the Asserted ‘722 Claims. Further, to the extent that any of the Asserted ‘722 Claims are construed sufficiently broadly to cover one or more of the various accused products identified in IV’s infringement contentions, such claims are invalid pursuant to at least 35 U.S.C. § 112(a).

Southwest contends that the following terms of asserted claims 14, 16, 17, and 18 are indefinite under 35 U.S.C. § 112(b), and/or lack written description support under 35 U.S.C. § 112(a):

1. “live object recognizable by the client device” (Claim 14)

Indefiniteness: The term “live object” is not explicitly defined in the specification. It is unclear whether “live” refers to real-time data streaming, frequent updates, or simply any data object that can be modified. It is also unclear how the system determines if an object is “live.” Is it based on update frequency, metadata, or another criterion? Additionally, the phrase “recognizable by the client device” is subjective and lacks objective boundaries. Does “recognizable” require predefined software compatibility? Does it mean the client must be capable of identifying the object’s format, or merely receive and process it in some capacity? Without clear technical definitions, a POSA cannot ascertain the exact scope of this term, making the claim indefinite.

Written Description: The specification discusses update messages and object registration, but does not clearly define "live object." If "live object" is meant to be dynamic content, there should be clear criteria for distinguishing it from static content.

2. "gateway device ... configured to identify a category of the update message" (Claim 14)

Indefiniteness: The claim requires the gateway device to "identify a category" of an update message, but the patent does not define what "category" means. Are categories predefined or dynamically generated? Is "category" based on data type, source, priority, or some other factor? Without an objective standard for categorization, a POSITA cannot determine whether a system meets this limitation.

3. "determine a node type to which the identified category maps" (Claim 14)

Indefiniteness: "Node type" is vague and undefined. Mapping a category to a node type suggests a predefined relationship, but there is no indication of how these relationships defined, stored, or modified.

Written Description: The patent describes a hierarchy of nodes, but it does not specify how categories and node types are mapped.

4. "cohesive respective application environments" (Claim 14)

Indefiniteness: The phrase "cohesive respective application environments" is subjective. What makes an application environment "cohesive"? Does "cohesive" require synchronized updates, consistent data availability, or a particular processing order?

Written Description: The specification never explicitly defines what a "cohesive" application environment is.

C. U.S. Patent No. 7,949,785

Claims 30, 35, and 37 of the '785 Patent (herein, the "Asserted '785 Claims") are invalid under 35 U.S.C. §§ 112(a) and 112(b), respectively, for at least the reasons discussed below.

The Asserted '785 Claims contain subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the applicants, at the time the application was filed, had possession of the claimed invention. In addition, the Asserted '785 Claims contain subject matter that was not described in the specification in such a way as to enable one of ordinary skill in the art to make and/or use the invention. Further, the Asserted '785 Claims fail to particularly point out and distinctly claim the invention, such that the claim's scope cannot be ascertained with reasonable certainty.

To the extent that the Asserted '785 Claims are construed broadly to cover undisclosed embodiments, the claims are invalid because the '785 Patent specification does not enable a person skilled in the art to practice the full scope of the purported invention, nor does the '785 Patent specification show the applicants had within their possession the full scope of the Asserted '785 Claims. Further, to the extent that any of the Asserted '785 Claims are construed sufficiently broadly to cover one or more of the various accused products identified in IV's infringement contentions, such claims are invalid pursuant to at least 35 U.S.C. § 112(a).

Southwest contends that the following terms of asserted claims 30, 35, and 37 are indefinite under 35 U.S.C. § 112(b), and/or lack written description support under 35 U.S.C. § 112(a):

1. "virtual network manager" (Claim 30)

Indefiniteness: The term "virtual network manager" is not explicitly defined in the specification. Is it a physical device, a software system, or a combination of both? Is it a single entity or a collection of distributed components? How does it differ from existing network management systems?

Written Description: The specification discusses virtual networks and device registration, but does not clearly define or describe a "virtual network manager" as a distinct entity.

2. "receive a registration request from an agent associated with a device" (Claim 30)

Indefiniteness: The term "agent associated with a device" lacks clarity. What qualifies as an agent? How is association with the device determined?

3. "DNS server for the virtual network ... return a network address associated with a network route director" (Claim 30)

Indefiniteness: The phrase "network route director" is not well-defined. Is this a specific device, a logical function, or a dynamic routing protocol? How does it interact with standard routing mechanisms? The claim requires the DNS server to return multiple addresses, including **a** "network address" for the route director, a "private network address" for a second device, and a "virtual network address" for the second device. However, the patent does not specify these addresses are assigned, whether these addresses are static or dynamic, or how conflicts are resolved if multiple addresses exist.

Written Description: The specification does not adequately describe how a DNS server can return multiple types of addresses in a single response.

4. "maintain data to associate a virtual network address with a device in the virtual network" (Claim 37)

Indefiniteness: The term "maintain data" is ambiguous. What kind of data structure is used? Is this a persistent database, a temporary cache, or another storage method? Does this refer to active session tracking or long-term registration?

D. U.S. Patent No. 8,027,326

Claims 1, 4, and 18 of the '326 Patent (herein, the "Asserted '326 Claims") are invalid under 35 U.S.C. §§ 112(a) and 112(b), respectively, for at least the reasons discussed below.

The Asserted '326 Claims contain subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the applicants, at the time the application was filed, had possession of the claimed invention. In addition, the Asserted '326 Claims contain subject matter that was not described in the specification in such a way as to enable one of ordinary skill in the art to make and/or use the invention under 35 U.S.C. § 112(a). Further, the Asserted '326 Claims fail to particularly point out and distinctly claim the invention, such that the claim's scope cannot be ascertained with reasonable certainty under 35 U.S.C. § 112(b).

To the extent that the Asserted '326 Claims are construed broadly to cover undisclosed embodiments the claims are invalid because the '326 Patent specification does not enable a person skilled in the art to practice the full scope of the purported invention, nor does the '326 Patent specification show the applicants had within their possession the full scope of the Asserted '326 Claims. Further, to the extent that any of the Asserted '326 Claims are construed sufficiently broadly to cover one or more of the various accused products identified in IV's infringement contentions, such claims are invalid pursuant to at least 35 U.S.C. § 112(a).

Southwest contends that the following terms of asserted claims 1, 4, and 18 are indefinite under 35 U.S.C. § 112(b), and/or lack written description support under 35 U.S.C. § 112(a):

1. "plurality of data subcarriers" (Claims 1 & 18)

Indefiniteness: The term "plurality" lacks definitiveness as to the amount of data subcarriers and therefore lacks a clear quantitative or objective standard. What does "plurality" mean? Why do claims 1 and 18 first recite a "plurality of data subcarriers," then only refer to "data subcarriers" without the "plurality" aspect? Are they trying to quantitatively differentiate between the "plurality of data subcarriers," versus mere "data subcarriers"?

Written Description: The specification does not provide an explanation of what constitutes a "plurality" of data subcarriers.

2. "partially filling the frequency gap" / "partially fill the frequency gap" (Claims 1 & 18)

Indefiniteness: The term "partially filling/fill" lacks a clear quantitative or objective standard. What percentage of the frequency gap must be filled for it to be considered "partial"? Does "partially" refer to uniformly distributed subcarriers or only some regions being filled?

Written Description: The specification does not provide an explanation of what constitutes "partially" filling/filled in a way that distinguishes it from fully filling/filled.

3. "one or more guard bands" (Claims 1 & 18)

Indefiniteness: The phrase "one or more guard bands" is vague. If a system uses multiple guard bands or different sizes of guard bands, does the claim apply to all configurations? Without defining the minimum or maximum required spacing, the claim scope is unclear.

Written Description: The specification does not clearly define how the guard bands are formed or constrained. If guard bands are fundamental to the claimed invention, they should be described with specific examples or constraints.

4. "using full spectral synthesis capability of a fast Fourier transform or an inverse fast Fourier transform" (Claims 1 & 18)

Indefiniteness: The phrase "full spectral synthesis capability" is ambiguous. Does "full" mean that all available subcarriers must be utilized? Does it require a specific mathematical implementation? If different FFT/IFFT algorithms process data differently, how do we determine what qualifies as "full spectral synthesis capability"?

5. "combining the first channel and the second channel using channel bonding with orthogonal frequency division multiplexing (OFDM)" (Claims 1 & 18)

Indefiniteness: The term "combining" is broad and not clearly defined. Does it mean concatenation of two channels? Does it mean joint encoding of data across both channels? Can the method use adaptive or dynamic bonding, or must it be static?

6. "transmitting the first channel and the second channel comprises transmitting to multiple radios" (Claim 4)

Indefiniteness: The phrase "transmitting to multiple radios" is unclear. Does it mean simultaneous transmission to all radios, or can transmission be sequential? Must each radio receive data from both channels, or can one radio receive data from only one channel?

Written Description: The specification does not describe how the first channel and second channel transmit to "multiple radios."

E. U.S. Patent No. 7,324,469

Claims 24, 25, 26, and 32 of the '469 Patent (herein, the "Asserted '469 Claims") are invalid under 35 U.S.C. §§ 112(a) and 112(b), respectively, for at least the reasons discussed below.

The Asserted '469 Claims contain subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the applicants, at the time the application was filed, had possession of the claimed invention. In addition, the Asserted '469 Claims contain subject matter that was not described in the specification in such a way as to enable one of ordinary skill in the art to make and/or use the invention. Further, the Asserted '469 Claims fail to particularly point out and distinctly claim the invention, such that the claim's scope cannot be ascertained with reasonable certainty.

To the extent that the Asserted '469 Claims are construed broadly to cover undisclosed embodiments, the claims are invalid because the '469 Patent specification does not enable a person skilled in the art to practice the full scope of the purported invention, nor does the '469 Patent specification show the applicants had within their possession the full scope of the Asserted '469 Claims. Further, to the extent that any of the Asserted '469 Claims are construed sufficiently broadly to cover one or more of the various accused products identified in IV's infringement contentions, such claims are invalid pursuant to at least 35 U.S.C. § 112(a).

Southwest contends that the following terms of asserted claims 24 and 32 are indefinite under 35 U.S.C. § 112(b), and/or lack written description support under 35 U.S.C. § 112(a):

1. An "Internet Hotspot"

Indefiniteness: to the extent the preamble is found to be limiting, it is unclear whether an "Internet Hotspot" is intended to exclude any manner of providing wired or wireless internet access to one or more users. In particular, the specification states that "[t]he present invention provides rural 'Hotspots' (***s**uch as Wi-Fi access, for example*) to enable wireless and hard wired, satellite distributed Internet access for anyone with a PC or other web-ready device (wireless ready or cabled) and a valid credit card." '469 patent at 1:34-37 (emphasis added).

Written Description: to the extent the preamble is found to be limiting, the specification fails to define a "Hotspot" or distinguish a "Hotspot" from any other manner of providing "Wi-Fi access, for example."

2. "subscriber access unit" / "the subscriber access unit being capable of authenticating a subscription account associated with a user prior to allowing the user access to the Internet" (Claim 24)

Indefiniteness: This term is indefinite, as the patent makes no attempt to define a "subscriber access unit" other than its "being capable of authenticating a subscription account associated with a user prior to allowing the user access to the Internet." The specification contains

contradictory descriptions of "authentication," including that "account status is *first* authenticated by the remote server" but that "*authentication of the user's account is performed by a subscriber merchant service*," in which case the described remote subscriber access unit is not "capable of authenticating a subscription account" as recited in the claim. '469 2:9-10 (emphasis added); *id*. at 2:33-34 ("authentication of the user's account is performed by a subscriber merchant service"). The specification's occasional use of capitalization indicates the patentee intended this to be a defined term, but failed to include its definition. *E.g.*, 4:2-3.

Written Description: The specification provides no description of how the subscriber access unit operates to be "capable of authenticating a subscription account associated with a user prior to allowing the user access to the Internet," by example or otherwise.

3. "operatively coupled to," "operatively coupled between" (Claim 24)

Indefiniteness: As used in the patent, the term "operatively coupled" is vague and ambiguous and thus indefinite. In particular, for example, the specification provides that "[a]s used herein, the term 'operatively coupled' refers to any configuration in which two or more devices *interact in any way*, by wired or wireless connections." '469 patent at 4:25-28 (emphasis added). A person of ordinary skill in the art would be unable to determine the scope of the invention, particularly whether any order or configuration of recited components is required, including whether intervening components would render a system within the scope or not.

Written Description: The limitation lacks written description to enable one skilled in the art to practice the invention, instead confirming "operatively coupled" or "operatively coupled between" have no meaning at all as used in the patent.

4. "a web-ready device operatively coupled to the at least one router" (Claim 24)

Indefiniteness: This term is indefinite as used in the claim and in view of the Plaintiff's Preliminary Infringement Contentions, which indicate that "enabling" end-user devices to access the Internet is sufficient to provide, as the claim requires, a "web-ready device."

Written Description: The limitation lacks written description because the specification merely repeats the term "web-ready device" without description.

5. "a remote location experiencing a relatively high volume of transient traffic" (Claim 24)

Indefiniteness: This limitation is indefinite because a person skilled in the art cannot ascertain with reasonable certainty the scope of what is "a remote location," "a relatively high volume" or "transient traffic." The specification provides the examples of "rest areas, restaurants, truck stops, rural hotels, conference centers, motels and state park lodges" ('469 patent at 1:41-44), all of which can and do have wildly different volumes of traffic and wildly different relation to the term "transient." For example, a conference center may house upwards of thousands of attendees who remain in the location for days on end. A truck stop might see anywhere from a handful to several hundred visitors, but for only brief periods of time during rest periods on trucking routes. A restaurant can be located in a busy shopping mall or in a quaint hamlet. Yet all are examples in the specification of "remote location[s] experiencing a relatively high volume of transient traffic." The specification provides no hints as to how remote is "remote," how many users makes a "relatively high volume," and how much or little time connected is sufficient to constitute "transient traffic."

Written Description: The specification provides no description of "a remote location experiencing a relatively high volume of transient traffic" except by inconsistent examples, instead repeatedly reciting only the claim language for this term.

6. "the data connection is one of a wired data connection and a wireless data connection" (Claim 25)

Indefiniteness: This claim is indefinite in view of the specification and claim 24, in which a "user may authenticate the Subscription account and access the Internet at the remote location by establishing **a** data connection," singular. It is unclear whether claim 25 requires the simultaneous use of wired and wireless connections by the user for the same data connection of claim 24. If claim 25 does not refer to "a data connection" recited in claim 24, it lacks antecedent basis.

Written Description: The specification does not describe the use of "a wired data connection and a wireless data connection" for a single data connection as recited by claim 25 and in view of independent claim 24.

7. "the wireless connection is one of an 802.11a wireless area network, an 802.11b wireless area network, an 802.11g wireless area network, and an 802.11n wireless area network" (Claim 32)

Indefiniteness: This claim is indefinite for lack of antecedent basis insofar as "the wireless connection" lacks antecedent basis; claim 25 recites "wherein the data connection is one of a wired data connection and a wireless data connection," not a "wireless connection." It is unclear whether "wireless connection" and "wireless data connection" are intended to be interchangeable, and, if so, how a simultaneously wired and wireless data connection can comprise "one of an 802.11a wireless area network . . ."

Written Description: The specification lacks written description of an "802.11n wireless area network" at least insofar as the 802.11n standard had not yet been drafted by the time of the filing of the provisional application giving rise to the '469 patent. *See supra* VIII.2.

F. U.S. Patent No. 7,257,582

Claims 1, 2, 5, and 7 of the '582 Patent (herein, the "Asserted '582 Claims") are invalid under 35 U.S.C. §§ 112(a) and 112(b), respectively, for at least the reasons discussed below.

The Asserted '582 Claims contain subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the applicants, at the time the application was filed, had possession of the claimed invention. In addition, the Asserted '582 Claims contain subject matter that was not described in the specification in such a way as to enable one of ordinary skill in the art to make and/or use the invention. Further, the Asserted '582 Claims fail to particularly point out and distinctly claim the invention, such that the claim's scope cannot be ascertained with reasonable certainty.

To the extent that the Asserted '582 Claims are construed broadly to cover undisclosed embodiments the claims are invalid because the '582 Patent specification does not enable a person skilled in the art to practice the full scope of the purported invention, nor does the '582 Patent specification show the applicants had within their possession the full scope of the Asserted '582 Claims. Further, to the extent that any of the Asserted '582 Claims are construed sufficiently broadly to cover one or more of the various accused products identified in IV's infringement contentions, such claims are invalid pursuant to at least 35 U.S.C. § 112(a).

Southwest contends that the following terms of asserted claims 1, 2, 5, and 7 are indefinite under 35 U.S.C. § 112(b), and/or lack written description support under 35 U.S.C. § 112(a):

1. "automatically determining file allocation and logically subdividing records of said input file into a plurality of partitions" (Claim 1)

Indefiniteness: The term "automatically determining file allocation" is vague and undefined. What criteria define "automatic" allocation? Is the allocation based on file size, content, metadata, or another heuristic? Can manual intervention be involved at any stage and still meet

this limitation? The phrase "logically subdividing records" is also ambiguous. What constitutes a logical subdivision? Does the subdivision require a specific algorithm or data structure? If different methods (e.g., fixed-size vs. content-based) could be used, which ones fall under the claim scope?

2. "distributing descriptions of all of said partitions to each of a plurality of subtask processors" (Claim 1)

Written Description: The specification does not provide examples or implementations for how "descriptions" are structured or distributed.

3. "repeating step (c) in at least some of the subtask processors each with another unprocessed partition on a first-come/first-served basis" (Claim 1)

Indefiniteness: The phrase "first-come/first-served basis" is undefined. The specification does not appear to describe how a first-come/first-served scheduling policy is implemented.

4. "generating at least one output combining all of the subtask outputs and reflecting the processing of all of said subtasks" (Claim 1)

Written Description: The specification does not provide a detailed method for combining outputs (e.g., sorting, appending, aggregating), nor does it address whether outputs must be stored in a specific format or can remain in multiple files.

5. "wherein the output in step (e) is an accumulation of output records from said subtasks in an arbitrary order" (Claim 5)

Indefiniteness: The phrase "arbitrary order" is ambiguous. Does it mean randomized order? Just non-sorted? If two identical executions produce different orderings, does that still meet the claim limitation?

XI. GROUNDS OF INVALIDITY BASED ON OBVIOUSNESS-TYPE DOUBLE PATENTING

A. Obviousness-Type Double Patenting of the '844 Patent in View of U.S. Patent No. 7,721,282

Southwest provides further notice to Plaintiffs that the asserted claims of the '844 Patent are invalid under the doctrine of obviousness-type double patenting ("OTDP") in view of claim 15 of U.S. Patent No. 7,721,282 (the "'282 Patent"). Because the asserted claims of the '844 Patent are directed to the same invention recited by claims presented in the '282 Patent, and because any differences in the asserted claims of the '844 Patent are, at most, obvious modifications of claims in the '282 Patent, the asserted '844 Patent claims are invalid. *See AbbVie Inc. v. Mathilda & Terrence Kennedy Inst. of Rheumatology Tr.*, 764 F.3d 1366, 1373 (Fed. Cir. 2014).

The '282 Patent is available as an OTDP reference against the '844 Patent. The '282 Patent was filed on March 30, 2006, issued on May 18, 2010, and is set to expire on December 19, 2027. The '844 Patent was filed on February 21, 2007, issued on December 11, 2012, and is set to expire on April 6, 2028. Thus, the '282 Patent was filed earlier than the '844 Patent, issued earlier than the '844 Patent, and is set to expire earlier than the '844 Patent.

As is the case here, where "the applicant chooses to file separate applications for overlapping subject matter . . . the doctrine of obviousness-type double patenting ensures that a particular invention (and obvious variants thereof) does not receive an undue patent term extension." *AbbVie*, 764 F.3d at 1373. Therefore, the earlier-filed, earlier-issued, and earlier-expiring '282 Patent qualifies as a double patenting reference.

Given that the '844 Patent and the reference '282 Patent share at least one common inventor and common ownership, determining whether a patent claim is invalid for OTDP entails two steps. *See id.* at 1374. The first step determines the differences between the respective claims of the two patents. The second step determines whether those differences render the later-expiring set of

claims sufficiently different to be patentably distinct. *Id.* "The second part of this analysis is analogous to the obviousness inquiry under 35 U.S.C. § 103 in the sense that if an earlier claim renders obvious or anticipates a later claim, the later claim is not patentably distinct and is thus invalid for obviousness-type double patenting." *UCB, Inc. v. Accord Healthcare, Inc.*, 890 F.3d 1313, 1323 (Fed. Cir. 2018) (citation omitted). The disclosure of the reference patent may be used to determine whether the claims merely define an obvious variation of what is disclosed and claimed in the reference patent. *Abbvie*, 764 F.3d at 1381.

The attached claim chart in Exhibit 844-OTDP-1 demonstrates that, to the extent there are any differences between the asserted claims of the '844 Patent and claims of the '282 Patent, these differences are not patentably distinct. The charts provide reasons why a person of ordinary skill in the art would conclude that the alleged invention defined in the asserted claims of the '844 Patent would have been an obvious variation of the alleged invention defined in the corresponding claim or claims of the '282 Patent.

B. Obviousness-Type Double Patenting of the '844 Patent in View of U.S. Patent No. 7,870,106

Southwest provides further notice to Plaintiffs that the asserted claims of the '844 Patent are invalid under the doctrine of obviousness-type double patenting ("OTDP") in view of claim 1 of U.S. Patent No. 7,870,106 (the "'106 Patent"). Because the asserted claims of the '844 Patent are directed to the same invention recited by claims presented in the '106 Patent, and because any differences in the asserted claims of the '844 Patent are, at most, obvious modifications of claims in the '106 Patent, the asserted '844 Patent claims are invalid. *See AbbVie Inc. v. Mathilda & Terrence Kennedy Inst. of Rheumatology Tr.*, 764 F.3d 1366, 1373 (Fed. Cir. 2014).

The '106 Patent is available as an OTDP reference against the '844 Patent. The '106 Patent was filed on February 2, 2006, issued on January 11, 2011, and is set to expire on May 18, 2027.

The ‘844 Patent was filed on February 21, 2007, issued on December 11, 2012, and is set to expire on April 6, 2028. Thus, the ‘106 Patent was filed earlier than the ‘844 Patent, issued earlier than the ‘844 Patent, and is set to expire earlier than the ‘844 Patent.

As is the case here, where “the applicant chooses to file separate applications for overlapping subject matter . . . the doctrine of obviousness-type double patenting ensures that a particular invention (and obvious variants thereof) does not receive an undue patent term extension.” *AbbVie*, 764 F.3d at 1373. Therefore, the earlier-filed, earlier-issued, and earlier-expiring ‘106 Patent qualifies as a double patenting reference.

Given that the ‘844 Patent and the reference ‘106 Patent share at least one common inventor and common ownership, determining whether a patent claim is invalid for OTDP entails two steps. *See id.* at 1374. The first step determines the differences between the respective claims of the two patents. The second step determines whether those differences render the later-expiring set of claims sufficiently different to be patentably distinct. *Id.* “The second part of this analysis is analogous to the obviousness inquiry under 35 U.S.C. § 103 in the sense that if an earlier claim renders obvious or anticipates a later claim, the later claim is not patentably distinct and is thus invalid for obviousness-type double patenting.” *UCB, Inc. v. Accord Healthcare, Inc.*, 890 F.3d 1313, 1323 (Fed. Cir. 2018) (citation omitted). The disclosure of the reference patent may be used to determine whether the claims merely define an obvious variation of what is disclosed and claimed in the reference patent. *Abbvie*, 764 F.3d at 1381.

The attached claim chart in Exhibit 844-OTDP-2 demonstrates that, to the extent there are any differences between the asserted claims of the ‘844 Patent and claims of the ‘106 Patent, these differences are not patentably distinct. The charts provide reasons why a person of ordinary skill in the art would conclude that the alleged invention defined in the asserted claims of the ‘844

Patent would have been an obvious variation of the alleged invention defined in the corresponding claim or claims of the '106 Patent.

Southwest reserves the right to assert any other double patenting challenges against any of the Asserted Patents.

XII. INVALIDITY CHARTS

Invalidity charts for each item of prior art are attached to these contentions as Exhibits 844-1 – 844-24, Exhibits 844-OTDP-1 – 844-OTDP-2, Exhibits 722-1 – 722-11, Exhibits 785-1 – 785-29, Exhibits 326-1 – 326-11, Exhibits 469-1 – 469-13, and Exhibits 582-1 – 582-6. These invalidity charts identify where in the prior art references each element of the asserted claim is found. While each element of each asserted claim is found in the identified prior art in multiple locations, the attached charts provide examples of citations sufficient to identify at least one such location where each claim limitation is found in each item of prior art. But the specific citations are only exemplary. Accordingly, Southwest and its expert(s) may rely on uncited portions of the prior art references as the citations must be interpreted in light of the entire disclosure of each reference. Additionally, because persons of skill in the art generally would appreciate an item of prior art in the context of other publications, literature, products, and understanding, Southwest and its expert(s) may rely on other publications and expert testimony as aids in understanding and interpreting the cited portions, for providing context to the cited portions, and as additional evidence that the prior art discloses a claimed feature. Southwest may establish what was known to a person having ordinary skill in the art through other publications, products, and/or testimony, and reserves their right to rely on cited and uncited portions of the prior art references, other publications, and/or testimony to establish that a person of skill in the art would have been motivated to combine the references rendering the claims obvious.

Southwest may modify its interpretation of the prior art based on additional analysis by Southwest's expert(s) and/or based on other circumstances that may affect the meaning or application of the claims.

XIII. DOCUMENT PRODUCTION

Southwest has produced the prior art references identified in these invalidity contentions. As noted herein, Southwest is also producing the below listed Exhibits with these invalidity contentions.

Exhibits	Description
844-1 – 844-24	Invalidity Charts for the '844 Patent
722-1 – 722-11	Invalidity Charts for the '722 Patent
785-1 – 785-29	Invalidity Charts for the '785 Patent
326-1 – 326-11	Invalidity Charts for the '326 Patent
469-1 – 469-13	Invalidity Charts for the '469 Patent
582-1 – 582-6	Invalidity Charts for the '582 Patent
844-OTDP-1 – 844-OTDP-2	Invalidity Charts Demonstrating Obviousness-Type Double Patenting for the '844 Patent

Date: April 8, 2025

Respectfully submitted,

/s/ S. Wallace Dunwoody

Michael C. Wilson
Texas Bar No. 21704590
mwilson@munckwilson.com
S. Wallace Dunwoody
Texas Bar No. 24040838
wdunwoody@munckwilson.com
MUNCK WILSON MANDALA, LLP
2000 McKinney Avenue, Suite 1900
Dallas, Texas 75201
(972) 628-3600
(972) 628-3616 fax

David G. Henry
Texas Bar. No. 09479355
dhenry@munckwilson.com
MUNCK WILSON MANDALA, LLP
510 Austin Avenue, Suite 3100
Waco, Texas 76701
(254) 362-2300

Tri T. Truong
Texas Bar No. 24102969
ttruong@munckwilson.com
MUNCK WILSON MANDALA, LLP
807 Las Cimas Parkway, Suite 300
Austin, Texas 78756
737-201-1600

Attorneys for Southwest Airlines Co.

CERTIFICATE OF SERVICE

I hereby certify that on April 8, 2025, a true and correct copy of Defendant Southwest Airlines Co.'s Preliminary Invalidity Contentions and accompanying exhibits were served via electronic mail to counsel of record.

/s/ S. Wallace Dunwoody
S. Wallace Dunwoody

EXHIBITS 844-1 – 844-24

Exhibit 844-1

Invalidity of U.S. Patent No. 8,332,844 in View of "Optimizing the Migration of Virtual Computers," Association for Computing Machinery, Special Interest Group on Operating Systems, Operating Systems Review, Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Volume 36, Issue S1 ("Sapuntzakis")

Sapuntzakis was published on December 31, 2002.[1] Sapuntzakis is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. § 102(b) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

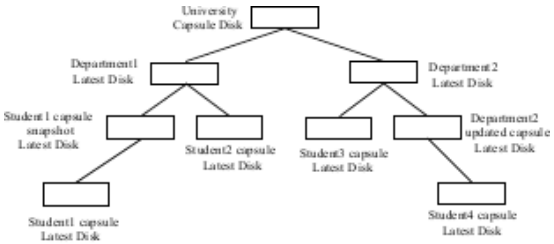
To the extent Plaintiff alleges Sapuntzakis does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Sapuntzakis or combine the teachings of Sapuntzakis with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

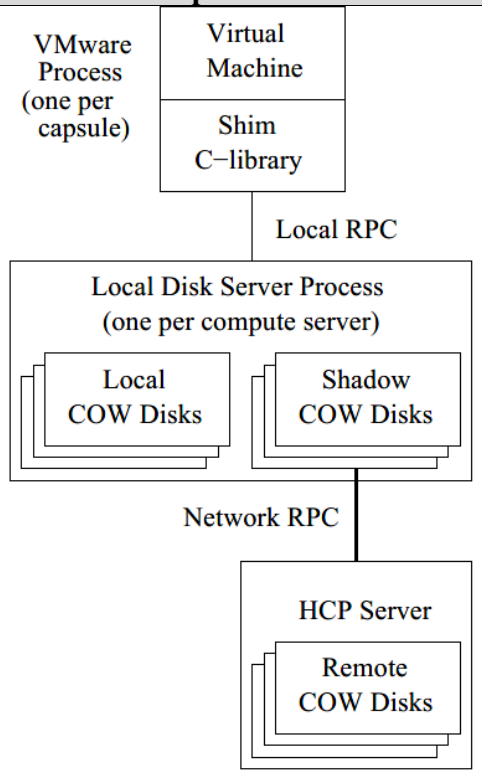
- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

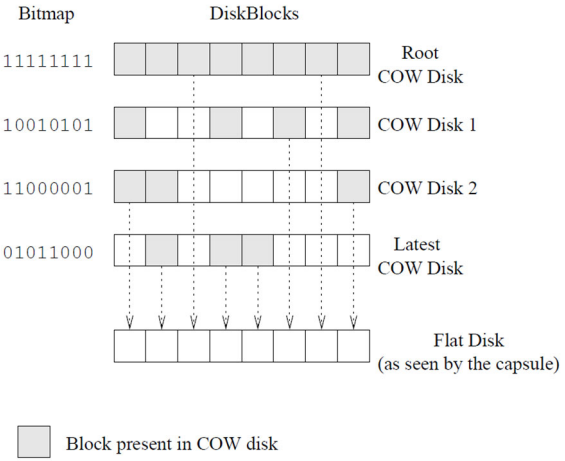
As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiffs' apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiffs' infringement contentions, or information obtained during discovery as the case proceeds.

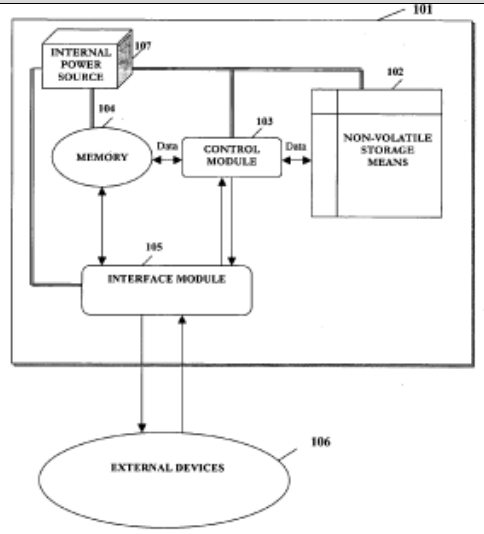
Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Sapuntzakis
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Sapuntzakis discloses: "Computers and storage in the Collective system act as caches of capsules. As users travel, the Collective can move their capsules to computers close to them, giving users a consistent environment. Capsules could be moved with users as they commute between home and work. Capsules can be duplicated, distributed to many different machines, and updated like any other data; this can form the basis for administering a group of computers." Sapuntzakis at 2.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"At the root of the hierarchy is a root disk, which is a complete capsule disk." Sapuntzakis at 5 "Figure 1 shows an example of a capsule hierarchy illustrating how it may be used in a university. The root capsule contains all the software available to all students." Sapuntzakis at 5.  Figure 1: An example capsule hierarchy. Sapuntzakis at 5.

Claim 7	Sapuntzakis
	 Figure 3: Implementation of capsule disks and demand paging. Sapuntzakis at 6.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes	"We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the

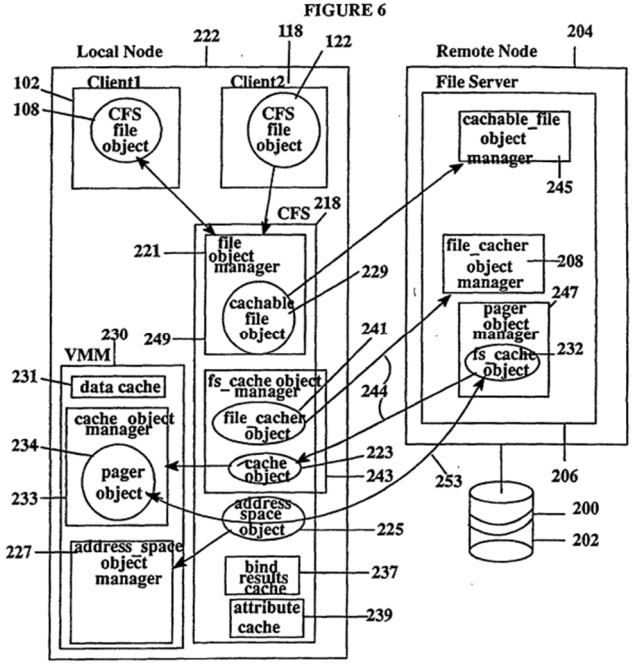
Claim 7	Sapuntzakis
do not include blocks of said root image that are unchanged by respective compute nodes; and	blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.  Figure 2: An example capsule disk. Sapuntzakis at 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"Each COW disk can either be local or remote. Each remote COW disk has a corresponding local shadow COW disk which contains all the locally cached blocks of the remote COW disk." Sapuntzakis at 6. "Reads, on the other hand, could either be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW." Sapuntzakis at 6.

Claim 7	Sapuntzakis
	"First, the memory in most systems caches disk blocks." Sapuntzakis at 7. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

Claim 7	Sapuntzakis
	 Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

Claim 7	Sapuntzakis
	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again

Claim 7	Sapuntzakis
	directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Sapuntzakis
	FIGURE 6  Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Sapuntzakis
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Sapuntzakis
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6.

Claim 11	Sapuntzakis
	"Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Kulkarni. Kulkarni concerns similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Kulkarni for example, to incorporate this limitation. *See, e.g.*, "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory

Claim 11	Sapuntzakis
	block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-2

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 7,089,300 ("Birse")

Birse titled "Method and apparatus for administering the operating system of a net-booted environment" to Cameron Stuart Birse et al. was filed on October 18, 1999 and issued on August 8, 2006.[1] Birse is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Birse does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Birse or combine the teachings of Birse with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Birse
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Birse discloses: "A method and apparatus are provided for supplying a reliable and maintainable operating system in a net-booted environment. According to one embodiment, a network computer (NC) system including an NC server and multiple NC clients is managed by an NC client causing the remainder of the NC clients that are subsequently booted to receive operating system software that is configured differently than that currently in effect by replacing one or more system volumes on the NC server containing the operating system software with one or more different system volumes." Birse at Abstract. "In order to obtain the benefits of low-cost administration, all storage of application programs and operating system software for an NC system is preferably on the NC server. This doesn't mean the NC clients have no mass storage, but rather that the NC clients boot over the network by accessing a stored copy of the operating system from the NC server. Additionally, when NC clients want to run an application, they access it from the NC server. Advantageously, in this manner, an application may be upgraded by simply replacing the old version of the application with a new version on the NC server. Then, the next time an NC client requests the application, the NC client will receive the new version." Birse at 4:53-64. *See also* Birse at 2:51-3:14.

Claim 7	Birse

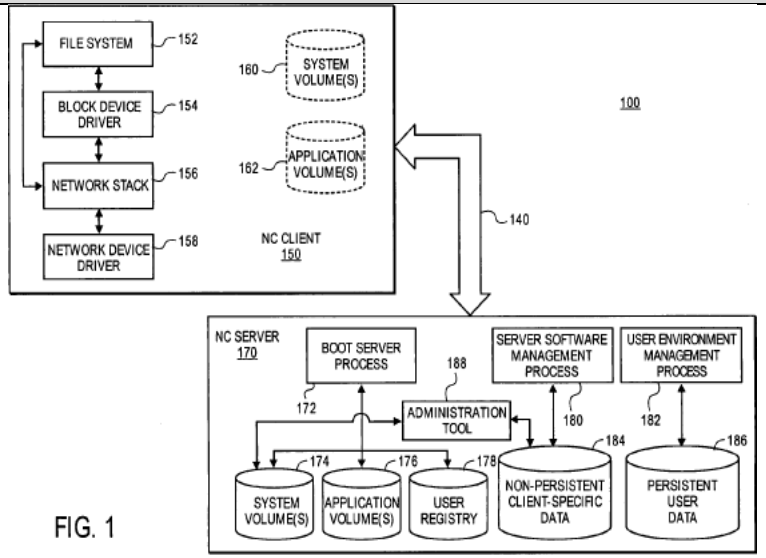


FIG. 1

Birse at FIG. 1

"FIG. 8 is a simplified block diagram illustrating two NC systems 810 and 820 coupled via the Internet 805."

Birse at 14:8-9.

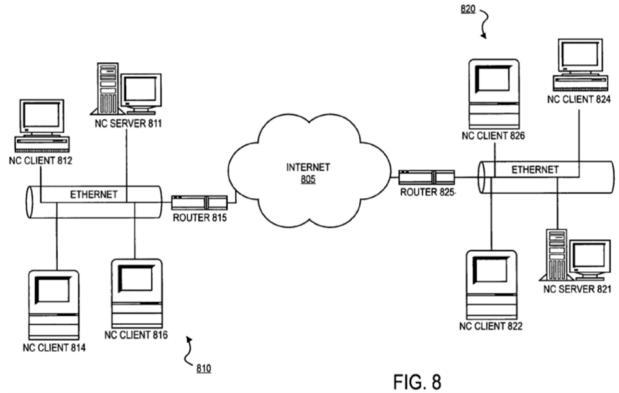
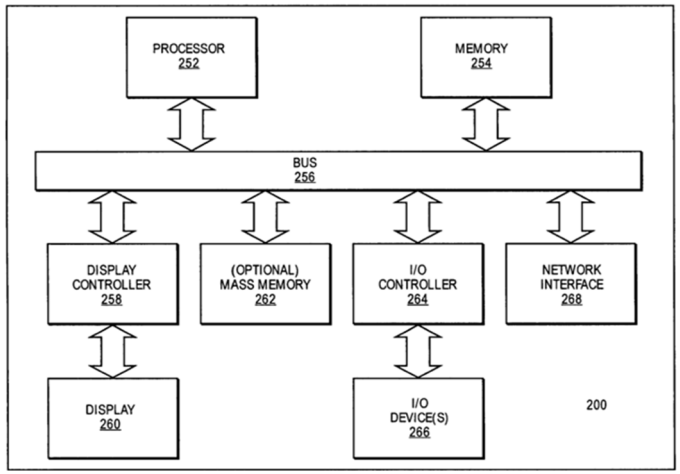


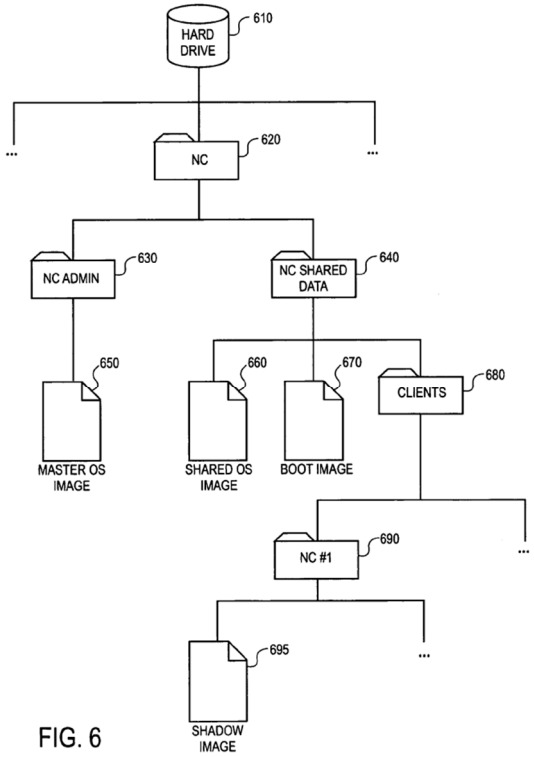
FIG. 8

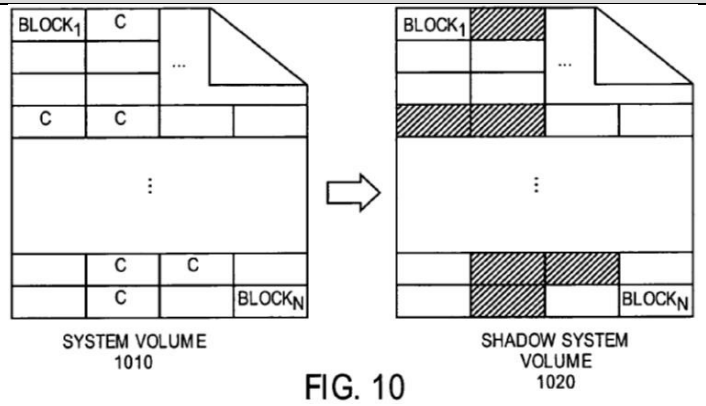
Birse at FIG. 8.

Claim 7	Birse
	"FIG. 2 is a block diagram of a digital processing system which may be used in accordance with one embodiment of the present invention. For example, the digital processing system 200 shown in FIG. 2 may be used as an NC client, an NC server, or other server computer system, such as an NT server. The digital processing system 200 may be interfaced to external systems through a network interface 268. It will be appreciated that the network interface 268 may be considered as part of the digital processing system 200. The network interface 268 may be an analog modem, an ISDN modem, a cable modem, a token ring interface, a satellite transmission interface, a wireless interface, or other interface(s) for providing a data communication link between two or more digital processing systems. The digital processing system 200 includes a processor 252, which may represent one or more processors and may include one or more conventional types of such processors, such as Motorola PowerPC processor, an Intel Pentium (or x86) processor, etc. A memory 254 is coupled to the processor 252 by a bus 256. The memory 254 may be a dynamic random access memory (DRAM) and/or may include static RAM (SRAM). The processor may also be coupled to other types of storage areas/memories (e.g., cache, Flash memory, disk, etc.), which could be considered as part of the memory 254 or separate from the memory 254. The bus 256 further couples the processor 252 to a display controller 258, an optional mass memory 262, the network interface 268, and an input/output (I/O) controller 264. The mass memory 262 may represent a magnetic, optical, magneto-optical, tape, and/or other type of machine-readable medium/device for storing information. For example, the mass memory 262 may represent a hard disk, a read-only or writeable optical CD, etc. The display controller 258 controls in a conventional manner a display 260, which may represent a cathode ray tube (CRT) display, a liquid crystal display (LCD), a plasma display, or other type of display device. The I/O controller 264 controls I/O device(s) 266, which may include one or more keyboards, mouse/trackball or other pointing devices, magnetic and/or optical disk drives, printers, scanners, digital cameras, microphones, etc." Birse at 7:63-8:35.

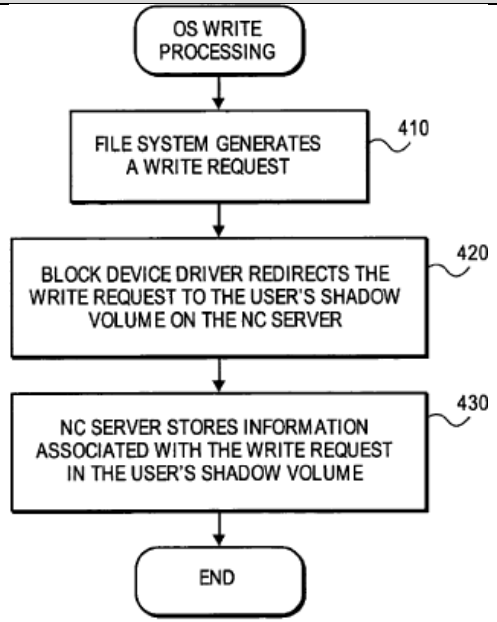
Claim 7	Birse
	 FIG. 2 Birse at FIG. 2.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"The NC server 170 includes one or more system volumes 174, one or more application volumes 176, a user registry 178, non-persistent client-specific data 184, and persistent user data 186. The system volumes 174 include a protected, read-only, master copy of the operating system software." Birse at 5:10-14. "FIG. 9 is a block diagram that conceptually illustrates NC client interaction with a SplitOS 920 according to one embodiment of the present invention. According to the example depicted, the SplitOS 920 of the NC server 170 contains a read-only core system volume image 922 and a read-write user system volume image 924. The core system volume 922 preferably contains those parts of the system that do not need to be written back to during system operation. One goal of the core system volume 922 is to provide all the system components that are mandatory for system operation." Birse at 14:27-34.

Claim 7	Birse
	"The user system volume 924 contains all the user-configurable system components, including preferences and all the system additions installed by application software, such as application-installed extensions and libraries. Additionally, the user system volume 924 may contain applications that cannot be run from an AppleShare server and/or other system components that do not function on a read-only volume." Birse at 14:41-44. "While for purposes of illustration, the boot server process 172, the server software management process 180, the user environment process 182, and the administration tool 188 are shown as part of a single NC server 170, in alternative embodiments, they may be distributed in whole or in part across multiple server computer systems. Additionally, the system volumes 174, the application volumes 176, the user registry 178, the non-persistent client-specific data 184, and the persistent user data 186 may be independently distributed across multiple server computer systems." Birse 6:59-63. "FIG. 6 is a simplified block diagram illustrating the layout of an exemplary hierarchical directory structure that may be used by an NC server according to one embodiment of the present invention. According to the embodiment depicted, a hard drive 610 of the NC server 170 or other mass storage device associated with the NC server 170 includes an NC folder (directory) 620. Within the NC folder 620 are an NC admin folder 630 and an NC shared data folder 640." Birse 12:30-38.

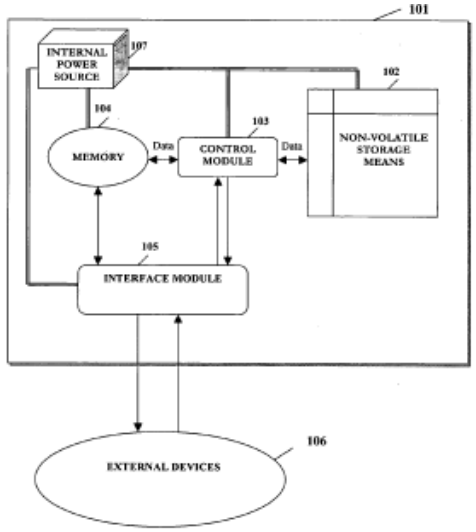
Claim 7	Birse
	 FIG. 6 Birse at FIG. 6. "The shared operating system image 660 is a read- write version of the read-only master operating system image 650 and may be mounted by the NC client 150 as part of step 395." Birse 12:44–47.

Claim 7	Birse
	 FIG. 10 Birse at FIG. 10.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"In this example, the NC clients are numbered from 1 to N and the clients folder 680 includes a folder for each NC client. For example, an NC #1 folder 690 is an area for storing client-specific information, such as a shadow image 695, corresponding to NC client #1." Birse 12:54–59. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write

Claim 7	Birse
	requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." Birse at 7:35-53; *see also* 11:30-32. "FIG. 4 is a flow diagram illustrating how NC client write requests directed to the system volumes 160 are handled according to one embodiment of the present invention." Birse at 10:63-65. "According to the embodiment depicted, at step 410, the file system 152 generates a write request directed to the system volumes 160. The write request is received by the block device driver 154 and redirected by the block device driver 154 at step 420 by translating it into a write request directed to the user's shadow volume on the NC server 170. At step 430, the NC server 170 stores information associated with the write request in the shadow volume associated with the NC client 150. In this manner, the file system 152 need not be aware that attempted modifications to the system volumes 160 are recorded instead in a client-specific shadow volume residing on the NC server 170." Birse at 11:6-17.

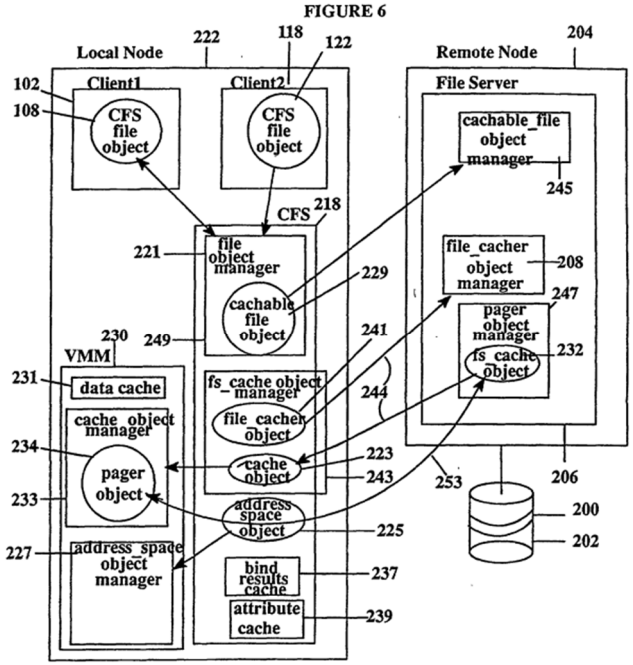
Claim 7	Birse
	 FIG. 4 Birse at FIG. 4. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52.

Claim 7	Birse
	"Preferably, the NC server 170 also creates a shadow system volume 930 for each connected NC client. The shadow system volume 930 shadows the user system volume 924 by storing modifications that are made to the user system volume 924." Birse 14:49-53. "As described above, according to one embodiment, the shadow system volume 930 is used by the block device driver 154 of the NC client 150 to implement a "copy-on-write" storage scheme in which modifications (writes) directed to the user system volume 930 are instead copied to the shadow system volume 930." Birse 14:56-61.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"The processor may also be coupled to other types of storage areas/memories (e.g., cache, Flash memory, disk, etc.), which could be considered as part of the memory 254 or separate from the memory 254." Birse at 8:17-19. "It will be appreciated that the digital processing system 200 represents only one example of a system, which may have many different configurations and architectures, and which may be employed with the present invention. For example, Macintosh and Intel systems often have multiple busses, such as a peripheral bus, a dedicated cache bus, etc." Birse at 8:36-41. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with

Claim 7	Birse
	Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].  Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the

Claim 7	Birse
	component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module." Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the

Claim 7	Birse
	fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Birse
	 Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Birse
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Birse
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67.

Claim 11	Birse
	"When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Kulkarni. Kulkarni concerns similar subject matter as this reference,

Claim 11	Birse
	namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Kulkarni for example, to incorporate this limitation. *See, e.g.*, "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory."

Claim 11	Birse
	Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-3

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2002/0034105 ("Kulkarni")

Kulkarni titled "Systems and methods for incrementally updating an image in flash memory" to Harish Kulkarni et al. was filed on January 8, 2001 March 21, 2002.[1] Kulkarni is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Kulkarni does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Kulkarni or combine the teachings of Kulkarni with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

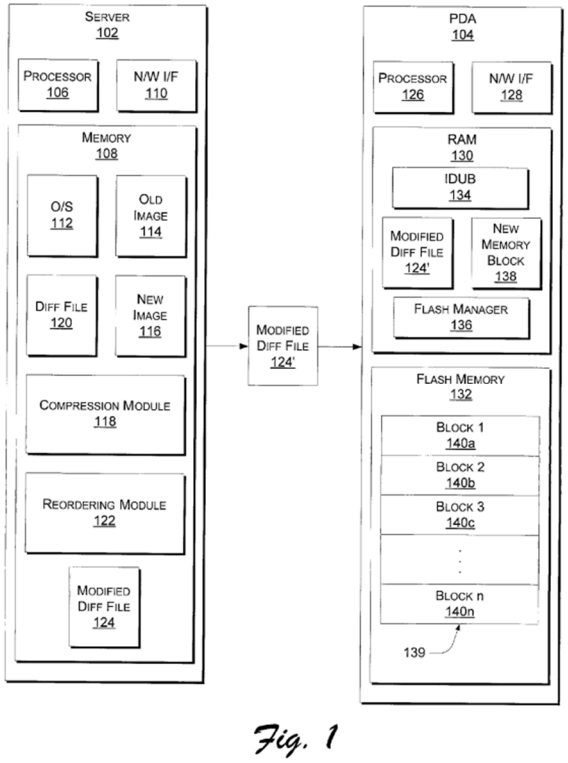
- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson"); and/or
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

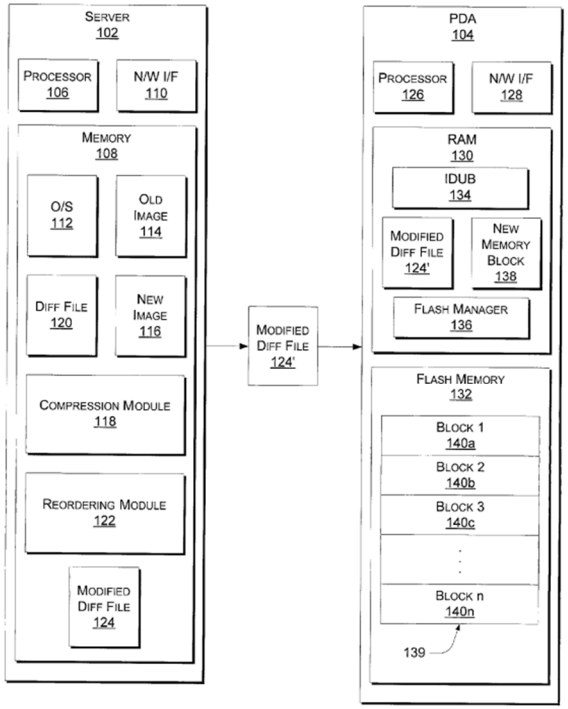
[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

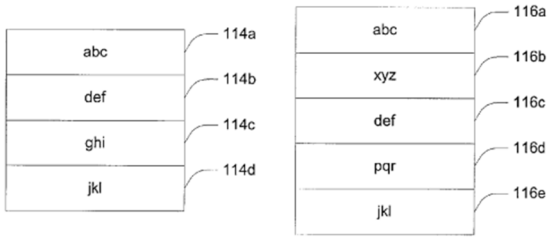
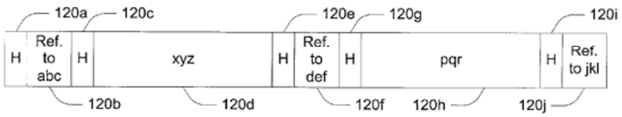
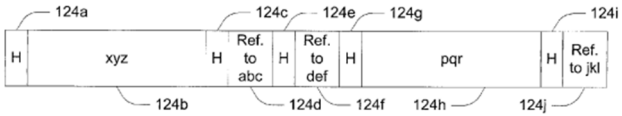
Claim 7	Kulkarni
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Kulkarni discloses "FIG. 1 is a system 100 implemented to incrementally update an image in flash memory. The system 100 includes a server 102 and a personal digital assistant (PDA) 104. The PDA 104 may be any type of electronic computing device that has flash memory and the capability to update an image in the flash memory." Kulkarni at [0026]. "The server 102 includes a processor 106, memory 108 and a network interface 110 for communicating with remote computing devices over a network, such as the Internet. The memory 108 includes an operating system 112, an old image 114 and a new image 116. The old image 114 is an execute-in-place (XIP) image to be stored in and run from flash memory. The old image 114 may be any image that can be stored in flash memory, but for purposes of the present discussion, the old image 114 is an operating system for the PDA 104." Kulkarni at [0027].

Claim 7	Kulkarni
	 *Fig. 1* Kulkarni at FIG. 1.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"The memory 108 also includes a compression module 118 configured to create a differences file 120 from the old image 114 and the new image 116. The differences file 120 is transmitted by the server 102 to the PDA 104. Therefore, compressing the differences file 120 prior to transmitting it to the PDA 104 significantly reduces the time required to transmit the differences file 120 to the PDA 104. The compression module 118 constructs the differences file 120 by including the new data sections contained in the new image 116. For the data sections in the new image 116 that are also contained in the old image 114, the compression module 118 includes references in the differences file 120 to the data

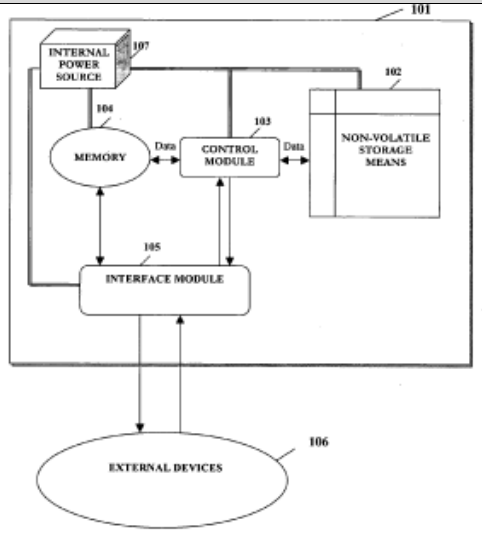
Claim 7	Kulkarni
	sections in the old image 114. Since a reference is typically much smaller than a data section it references, the differences file 120 is significantly smaller than the new image 116." Kulkarni at [0029].
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"As this discussion progresses, it will be seen that a new image is constructed from the differences file and the old image, overwriting the old image as it is built. Therefore, if a dependent section requires data from a referenced section from the old image that has previously been overwritten, then it will be impossible to construct the dependent section." Kulkarni at [0031].

Claim 7	Kulkarni
	 *Fig. 1* "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the

Claim 7	Kulkarni
	server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037]. "The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. "FIG. 2 c depicts the differences file 120 that is obtained from the old image 114 and the new image 116. A header 120 a is located at the beginning of the differences file 120. Section 120 b follows header 120 a and contains a reference to section 114 a ('abc') in the old image 114. Since the new image 116 contains a section of data identical to section 114 a ('abc'), then a reference may be included and the section having data 'abc' may be taken from the old image 114 when building the new image 116 on the PDA 104." Kulkarni at [0041].

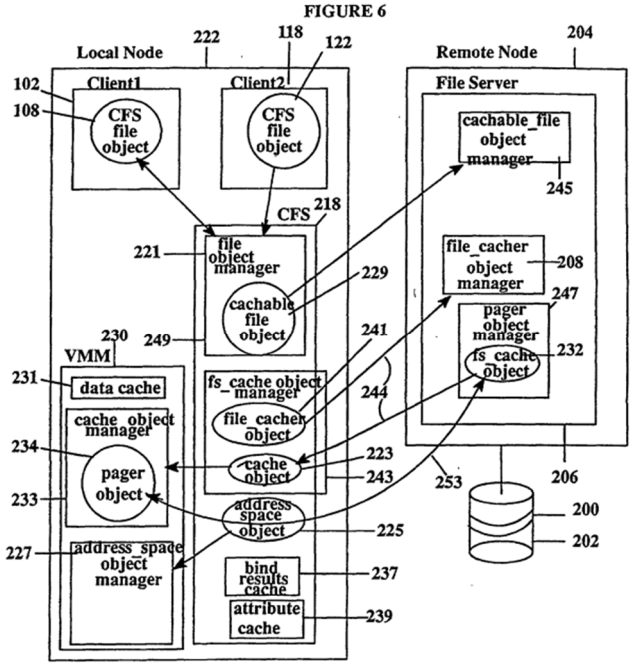
Claim 7	Kulkarni
	 OLD IMAGE 114 Fig. 2a NEW IMAGE 116 Fig. 2b  DIFFERENCES FILE 120 Fig. 2c  MODIFIED DIFFERENCES FILE 124 Fig. 2d Kulkarni at FIGS. 2a-2d.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	One solution to the problem of cyclic references is to temporarily buffer both data sections in the RAM 130, then write both the sections to the flash memory 132 at the same time. This solution may work for a simple cycle consisting of only two data sections, but a cycle can consist of any number of data sections. If a large

Claim 7	Kulkarni
	number of data sections are cached in the RAM 130, then the aim of compression and incremental construction is defeated. Kulkarni at [0053]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

Claim 7	Kulkarni
	 Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

Claim 7	Kulkarni
	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object

Claim 7	Kulkarni
	manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Kulkarni
	 Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Kulkarni
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Kulkarni
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory."

Claim 11	Kulkarni
	Kulkarni at [0013]. "FIG. 6 illustrates the process performed on the PDA 104 to incrementally update the image 139 in flash memory 132. At step 600, the PDA 104 boots into the incremental dial-up bootloader 134 and dials into the server 102 to download the modified differences file 124. The modified differences file 124′ is downloaded to the PDA 104 and stored in the RAM 130 at step 402." Kulkarni at [0068]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069].

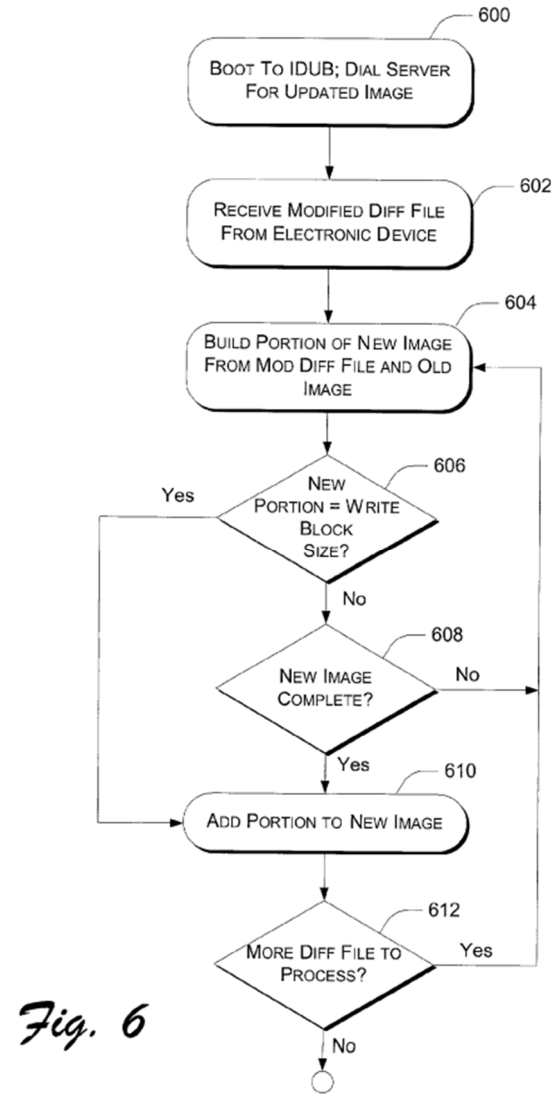
Claim 11	Kulkarni
	 *Fig. 6* Kulkarni at FIG. 6.

Exhibit 844-4

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2005/0004925 ("Stahl")

Stahl titled "Copy-on-write mapping file system" to Nathaniel Stahl et al. was filed on May 7, 2004 and published on January 6, 2005.[1] Stahl is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Stahl does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Stahl or combine the teachings of Stahl with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

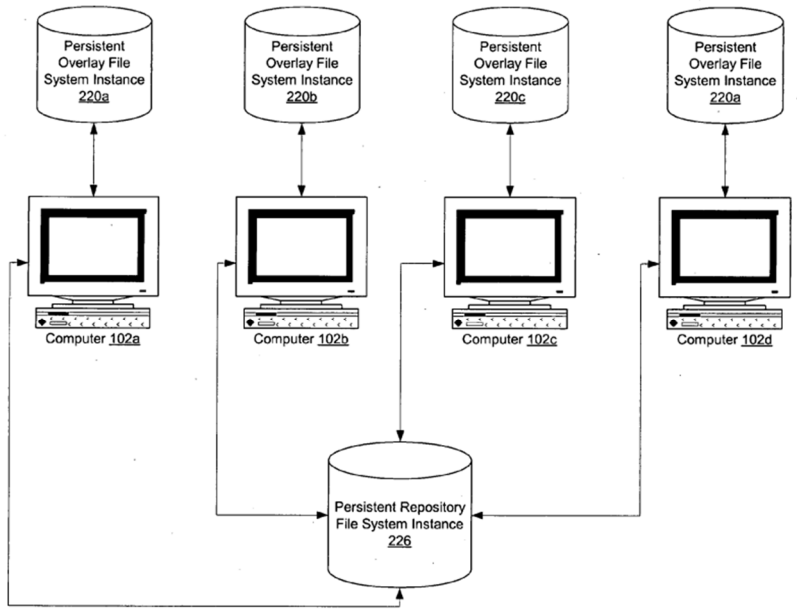
- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

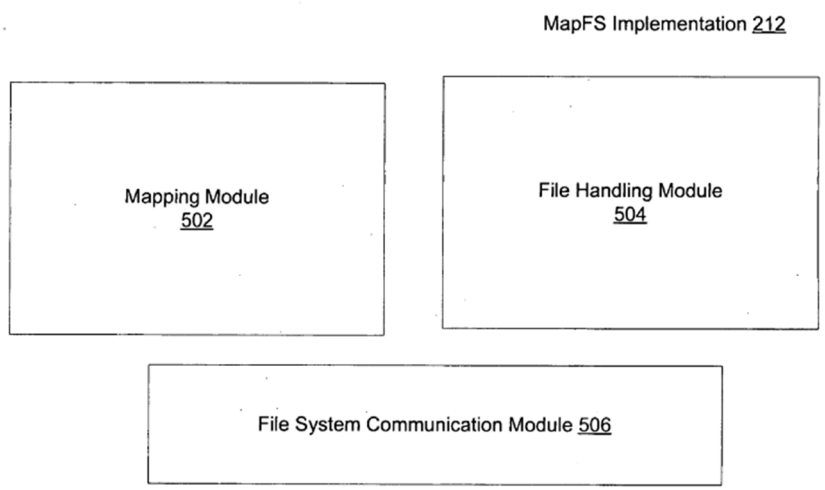
Claim 7	Stahl
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Stahl discloses "A system provides sharing of read-only file systems while at the same time providing each client of the read-only file system the ability to write to its own data store. Files can be either on a read-only persistent repository file system, or on a writeable persistent overlay file system. An "optimistic sharing" paradigm means that by default, everything on the file system is assumed to be read-only. If an attempt is made to modify a file—that is, a private copy is needed—the performance hit is typically minimal, because most written-to files are small. Even in the event of a larger file, the performance hit is a one-time cost." Stahl at Abstract. "The present invention enables "semi-sharing" of data. In general, this semi-sharing has application to environments in which a large volume of information is shared among many computer systems, but each computer system has a need to modify some portion of the data." Stahl at [0015]. In one embodiment, the present invention enables sharing of read-only file systems while at the same time providing each client of the read-only file system, e.g., a workstation, the ability to write to its own data store. Files can be either on a read-only persistent repository file system, or on a writeable persistent overlay file system. The present invention's "optimistic sharing" paradigm means that by default, everything on the file system is assumed to be read-only. If an attempt is made to modify a file—that is, a private copy is needed —the performance hit is typically minimal, because most written-to files are small. Even in the event of a larger file, the performance hit is a one-time cost. Stahl at [0016].
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"FIG. 1 illustrates conceptually a network architecture designed to take advantage of the present invention. FIG. 1 includes a number of computers 102, each

Claim 7	Stahl
	computer preferably including an instance of the Map File System as described below, and in communication with an instance of a read-only persistent repository file system 226. Each computer 102 also has read and write access to a persistent overlay file system 220. Thus, in a manner such as that described below, computers 102 can share a common read-only file store 226 while at the same time writing data as necessary to an overlay 220. Note that FIG. 1 illustrates only one of a variety of different ways of configuring a network using the described system. For example, repository file system 226 could be one or several physical drives. Similarly, overlay file system 220 may be one drive with multiple partitions and accessed by more than one computer 102, or may be one drive for each computer 102. As those of skill in the art will appreciate, the physical configuration can be any in which shared read-only access is given to the repository file system instance, and unshared write access is given to an overlay file system instance." Stahl at [0025].  *Fig. 1*

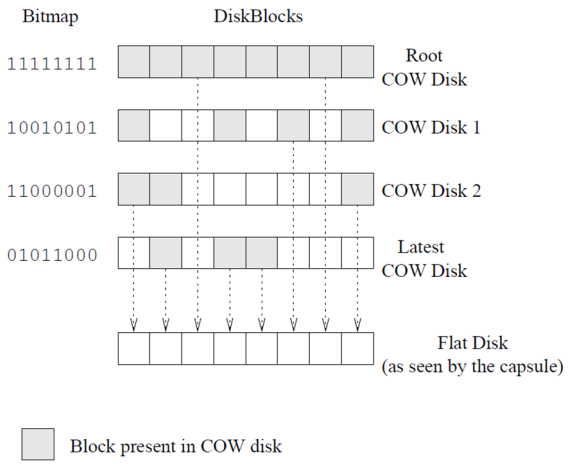
Claim 7	Stahl
	Stahl at FIG. 1. "FIG. 2 provides an overview of a preferred embodiment of a system 200 including a Map file system (MapFS) implementation 212. Syscall handler 202 is a conventional part of an operating system such as Linux, which receives and handles system calls from applications. For system calls that require interaction with an underlying file system, syscall handler 202 passes the request to a virtual file system layer (VFS layer) 204. VFS layer 204 is a conventional virtual file system layer such as one implemented by the Linux operating system. VFS layer 204 is an abstract layer designed to enable the upper layers of a system kernel to interact with different file systems. For example, the Linux VFS supports ext3, FAT and NFS file systems, as well as various others. The VFS layer allows programs to have a standard set of interfaces for managing files; allows the operating system and file system implementations to have a standard way of communicating with each other; and implements a library of common file system-related functions that file system implementations can use to avoid code duplication." Stahl at [0026]. "VFS Layer 204 is also in communication with MapFS Implementation 212. MapFS is a host file system that implements the interface expected by the VFS layer 204, with functionality described below. MapFS Implementation 212 presents an interface to the VFS Layer 204 that conforms to the VFS requirements of the particular operating system in use, e.g., Linux. The operation of MapFS Implementation 212 is further described below with respect to FIG. 5. A particular instance of a MapFS file system is MapFS Instance 216. The MapFS Instance 216 stores transient information for the MapFS file system as composed of information about the underlying view, repository and overlay file systems. In receiving requests from applications (via the VFS layer), passing the requests to repository and overlay file systems as appropriate, and satisfying the requests by returning a result to the application, MapFS provides the illusion to the requesting application that there is one coherent writeable file system." Stahl at [0027].

Claim 7	Stahl
	"Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "Overlay FS Implementation 214 is an implementation of a file system responsible for providing writeable storage on the persistent overlay file system instance 220. Overlay FS Implementation 214 receives and responds to file system requests—including write requests—from MapFS Implementation 212. Overlay FS Instance in memory 218 behaves analogously to Repository FS Instance 224. In a preferred embodiment, the Overlay FS is a directory tree rooted at a location specified when the MapFS is instantiated." Stahl at [0029].
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "Overlay FS Implementation 214 is an implementation of a file system responsible for providing writeable storage on the persistent overlay file system instance 220.

Claim 7	Stahl
	Overlay FS Implementation 214 receives and responds to file system requests—including write requests—from MapFS Implementation 212. Overlay FS Instance in memory 218 behaves analogously to Repository FS Instance 224. In a preferred embodiment, the Overlay FS is a directory tree rooted at a location specified when the MapFS is instantiated." Stahl at [0029]. "ViewFS Implementation 206 maintains a mapping between file names as referenced by MapFS Implementation 212 and as stored by the overlay and/or repository file systems. Persistent View FS Instance 210 maintains a physical (i.e. disk) record of the mappings, while ViewFS Instance 208 maintains information in memory similar to Repository FS Instance 224 and Overlay Instance 218. In a preferred embodiment, the View FS is a directory tree rooted at a location specified when the MapFS is instantiated. Stahl at [0030]. "In general, file requests received through VFS Layer 204 by MapFS Implementation 212 are satisfied by accessing a repository version of the file or by accessing an overlay version of the file. The view specifies which host file should be used to satisfy requests on the file. If the file is on a repository and an attempt is made to modify it, the file is migrated to the overlay. The following examples are illustrative." Stahl at [0030]. "FIG. 5 illustrates logical subunits that perform functionality of the MapFS Implementation 212 in one embodiment. MapFS Implementation 212 includes a mapping module 502, a file handling module 504 and a file system communication module 506. File system communication module 506 presents an interface for communicating with other file system implementations such as View FS implementation 206 and Repository FS implementation 222. Mapping module 502 provides mapping functionality for MapFS Implementation 212, including logic used to obtain locations of files or data structures within system 200. File handling

Claim 7	Stahl
	module 504 provides an interface for receiving and responding to requests from VFS layer 204 and additionally includes logic used to perform file system operations." Stahl at [0030].  *Fig. 5* Stahl at FIG. 5. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks

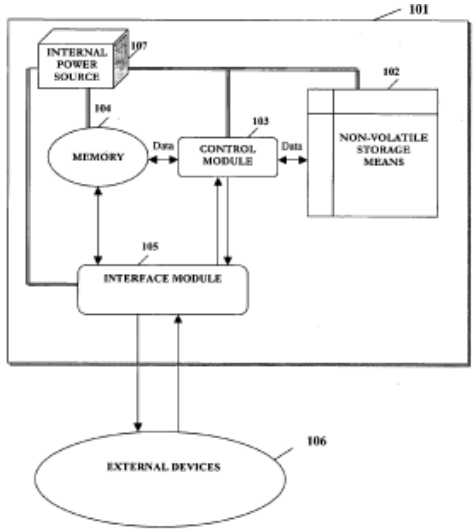
Claim 7	Stahl
	not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

Claim 7	Stahl
	 Figure 2: An example capsule disk. Sapuntzakis at 6. *See also* Sapuntzakis at 1-12. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally."

Claim 7	Stahl
	Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037]. "The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two

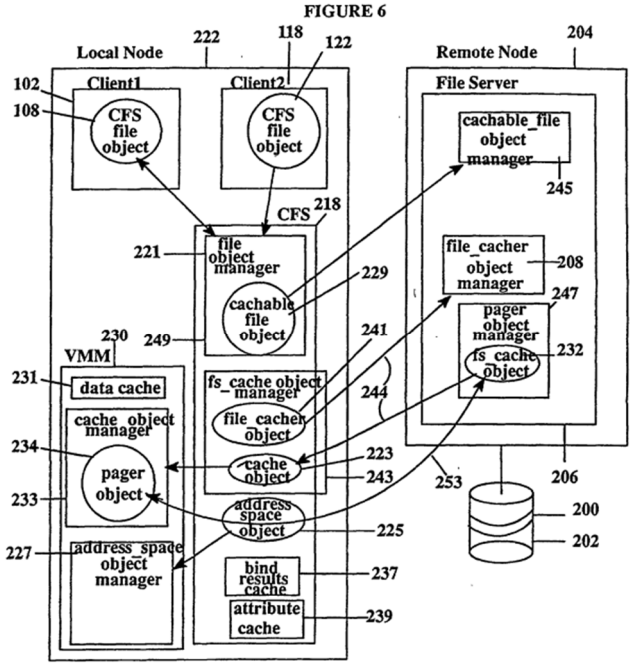
Claim 7	Stahl
	types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"In a system architecture contemplated by the present invention, one or more read-only persistent repository file systems are shared amongst many computers. Each computer has access to a writeable overlay file system. When an application executing on a computer attempts to write data to a file located on the read-only file system, the file is instead written to the overlay file system. Subsequent file operations on the file are directed towards the overlay file system instead of to the read-only file system. A mapping is maintained between filenames and their path locations, making the process transparent to the calling application. This eliminates the need for duplicate storage, thus saving money and improving performance of SAN and NAS devices by allowing more efficient use of disk caches. In addition, new servers can be deployed rapidly in response to changing load conditions, software updates, and the like." Stahl at [0017]. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view

Claim 7	Stahl
	caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. "Repository FS Implementation 222 reads 420 the contents of the file from the persistent repository FS instance on disk 226 and updates 422 the copy in memory, returning 424 the contents of the file to the MapFS Implementation 212. File handling module 504 writes 426 the data to the new file on the overlay by sending a write request to the Overlay FS Implementation 214. The Overlay FS Implementation 214 writes 428 the data to the Persistent Overlay FS Instance 226 and updates 430 the data cache in the Overlay FS instance in memory 218, returning 432 a success code to the MapFS Implementation 212." Stahl at [0044]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann and/or Nelson. Holzmann and Nelson concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's and/or Nelson's caching

Claim 7	Stahl
	teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann or Nelson for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].  Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the

Claim 7	Stahl
	component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module." Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind

Claim 7	Stahl
	operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Stahl
	 Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4.

Claim 11	Stahl
[11.0] The method as recited in claim 7 further comprising:	"Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to

Claim 11	Stahl
merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "Overlay FS Implementation 214 is an implementation of a file system responsible for providing writeable storage on the persistent overlay file system instance 220. Overlay FS Implementation 214 receives and responds to file system requests—including write requests—from MapFS Implementation 212. Overlay FS Instance in memory 218 behaves analogously to Repository FS Instance 224. In a preferred embodiment, the Overlay FS is a directory tree rooted at a location specified when the MapFS is instantiated." Stahl at [0029]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched."

Claim 11	Stahl
	Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver

Claim 11	Stahl
	154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As

Claim 11	Stahl
	previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-5

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2006/0168154 ("Zhang")

Zhang titled "System and method for a distributed object store" to Zheng Zhang et al. was filed on November 19, 2004 and published on July 27, 2006.[1] Zhang is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. § 102(e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Zhang does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Zhang or combine the teachings of Zhang with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

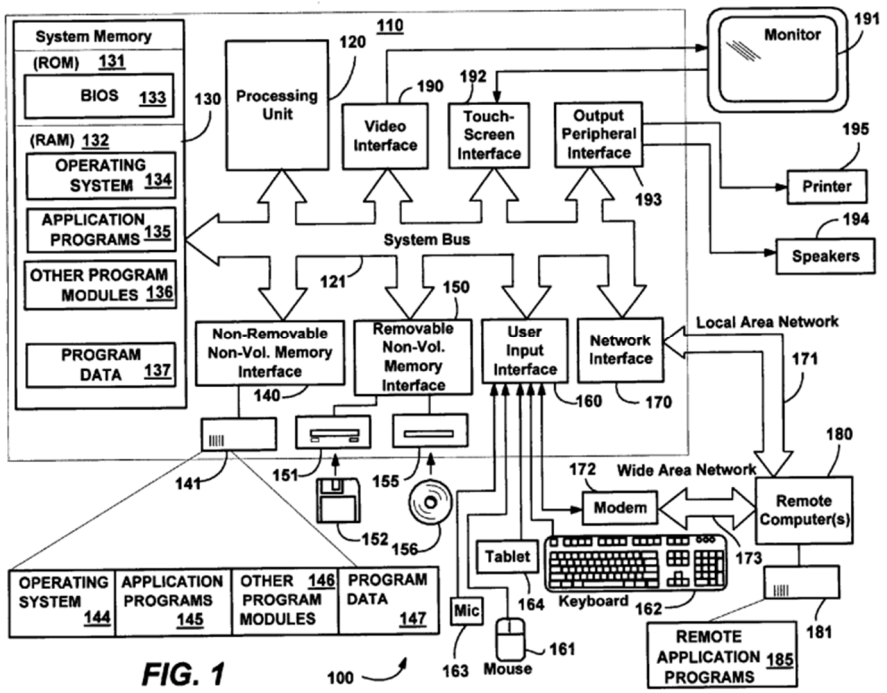
As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

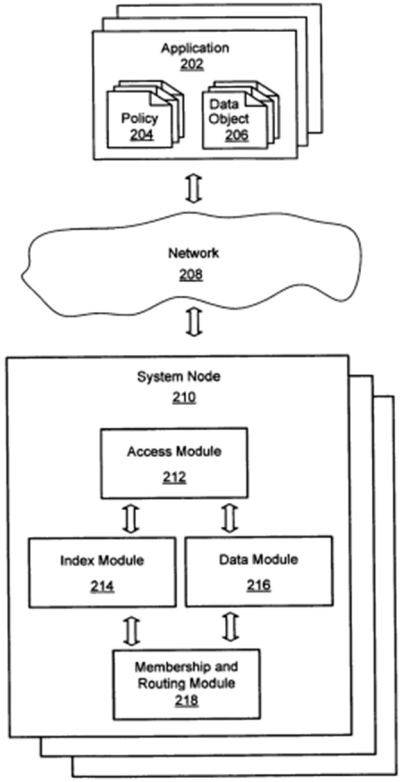
Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Zhang
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Zhang discloses: "An improved system and method for flexible object placement and soft-state indexing of objects in a distributed object store is provided. A distributed object store may be provided by a large number of system nodes operably coupled to a network. A system node provided may include an access module for communicating with a client, an index module for building an index of a replicated data object, a data module for storing a data object on a computer readable medium, and a membership and routing module for detecting the configuration of operable nodes in the distributed system. Upon failure of an index node, the failure may be detected at other nodes, including those nodes that store the replicas of the object. These nodes may then send new index rebuilding requests to a different node that may rebuild the index for servicing any access request to the object." Zhang at Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"Briefly, the present invention may provide a system and method for flexible object placement and soft-state indexing of objects in a distributed object store. A distributed object store may be provided by a large number of system nodes operably coupled to a network. A client executing an application may also be operably coupled to the network. A system node provided may include an access module for communicating with a client, an index module for building an index of a replicated data object, a data module for storing a data object on a computer readable medium, and a membership and routing module for detecting the configuration of operable nodes in the distributed system. A system node may be configured as an index node that may have an index module but not a data module. A system node may also be configured as a data node that may have a data module but not an index module." Zhang at [0008]. "With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110.

Claim 7	Zhang
	Components of the computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus." Zhang at [0025].  Zhang at FIG. 1.

Claim 7	Zhang
	"The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136 and program data 137." Zhang at [0027]. "The present invention is generally directed towards a system and method for a distributed object store. More particularly, the present invention provides object-level mirroring, flexible object placement and in memory indexing of objects in a distributed object store. Data object placement may flexibly support both random and application-specific placement policies. By using a soft-state approach for indexing, an index for a data object may be kept in main memory without needing to be replicated or persistently stored. In the event an index may be lost, then index repair may be driven by system nodes and the index may be rebuilt from the data objects stored on the system nodes. As long as there may be a last copy of the object available, the index rebuilding process may be initiated. Eventually the object may be repaired to the desired replication degree and the index may be rebuilt. Moreover, system nodes or storage containing data objects may be plugged into the distributed data network and their data objects will be automatically discovered and replicated. As will be understood, the various block diagrams, flow charts and scenarios described herein are only examples, and there are many other scenarios to which the present invention will apply." Zhang at [0031]. "Turning to FIG. 2 of the drawings, there is shown a block diagram generally representing an exemplary architecture of system components for a distributed

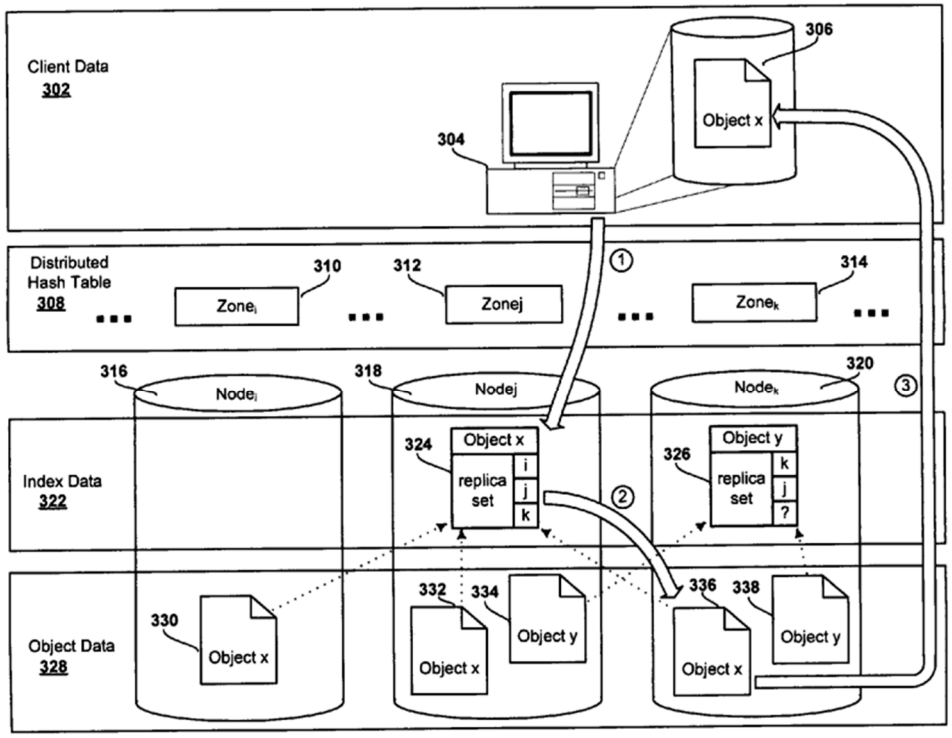
Claim 7	Zhang
	object store. Those skilled in the art will appreciate that the functionality implemented within the blocks illustrated in the diagram may be implemented as separate components or the functionality of several or all of the blocks may be implemented within a single component. For example, the functionality for the index module 214 may be included in the same component as the data module 216. Or the functionality of routing in the membership and routing module 218 may be implemented as a separate component." Zhang at [0032]. "As used herein, an object store may mean a large distributed system of operably coupled nodes. In various embodiments, one or more applications 202 may be operably coupled to one or more system nodes 210 by a network 208. In general, an application 202 may be any type of executable software code such as a kernel component, an application program, a linked library, an object with methods, and so forth. In one embodiment, an application may execute on a client computer or computing device, such as computer system environment 100 of FIG. 1 which may be operably coupled to one or more system nodes 210 by the network 208. An application 202 may include one or more policies 204 and one or more data objects 206. A policy 204 may generally include instructions for data object placement and replication in the distributed object store." Zhang at [0033].
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image	"Data object replicas may be placed on any system nodes. The replication degree, or number of replicas, may be specified for an individual data object and the replication degree may be different from object to object. A data object 206 may represent any type of data for storage in the distributed object store. For example, a data object may be any type of file(s), including a document, spreadsheet, data base, executable code and so forth. An object within the distributed object store may be immutable meaning the object may not be modified once placed into the distributed object store. In order to change an object, a new object may be created and the new object may be put into the object store. In one embodiment, the old object may be subsequently deleted."

Claim 7	Zhang
that are unchanged by respective compute nodes; and	Zhang at [0034].  *FIG. 2* Zhang at FIG. 2. "A system node 210 may be any type of computer system or computing device such as computer system environment 100 of FIG. 1. A system node 210 may include an access module 212, an index module 214, a data module 216, and a membership and routing module 218. Each of these modules may also be any type of executable software code such as a kernel component, an application program, a

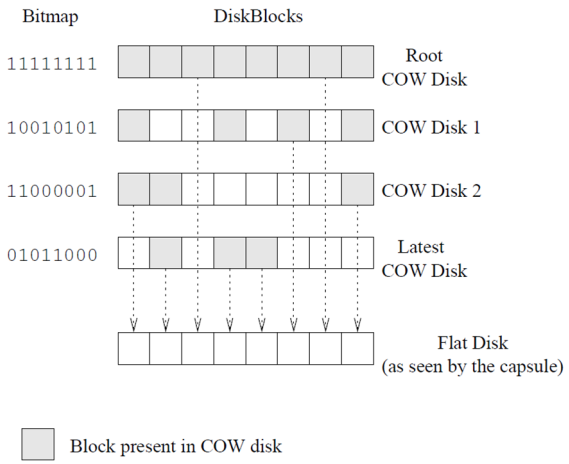
Claim 7	Zhang
	linked library, an object with methods, or other type of executable software code. A system node 210 containing an operable index module 214 shall be called an index node. A system node 210 containing an operable data module 216 shall be called a data node. A system node may be an index node, a data node, or both." Zhang at [0036]. "The access module may only be present on some system nodes and may be responsible for communicating with clients. The index module may also be only present on some system nodes and may be responsible for indexing a subset of data objects stored in the object store. The logical space of object identifiers may be partitioned into subsets and each index node may be responsible for indexing data objects whose identifiers belong to a given subset. For example, if the object identifiers are randomly chosen globally unique identifiers (GUIDs), then the subsets of object identifiers may be partitioned so that the data objects referenced by the object identifiers may be evenly distributed among the subsets of object identifiers. In this way, responsibility for tracking a large number of objects may be evenly spread among the index nodes." Zhang at [0037]. "An index built by an index module may include information about a replica set for each object such as an indication of system nodes where replicas of the object may be stored. For instance, an index may maintain a list of locations or pointers to all the copies of a data object. There may also be a replication degree defined for each object that specifies the number of copies of the object desired to be stored in the distributed object store. The replication degree may be stored as metadata associated to every copy of a data object. During some transient state or condition, there may be fewer copies than the number of copies specified by the replication degree due to node failure, in which case any copies of the data objects on failed nodes may no longer be available for access. It may be the responsibility of the index module to track such a situation and remedy it, for example, by directing creation of new copies until the desired number of available replicas may be stored in the object store."

Claim 7	Zhang
	Zhang at [0038]. "The data module 216 may also be only present on some system nodes and may be responsible for keeping track of what data objects may be stored on the computer storage media of that node. The data module may also keep track of the index nodes responsible for indexing those data objects as well. In the event that an index node may fail, data modules may be notified by the membership and routing module 218 and each data module may determine a new index node that may be responsible for indexing the data objects. In one embodiment, the data module may then report to that index module what data objects it should be indexing." Zhang at [0039]. "In one embodiment of the present invention, the identification space of data objects may be partitioned by the membership and routing module using a hash function that may create several zones, each of which may have a system node for indexing data objects assigned to that zone. A data object may be referred to herein as rooted at a particular index node if the partitioning of the identification space assigns the identifier of the data object to that index node." Zhang at [0045]. "FIG. 3 presents an exemplary illustration generally depicting a logical representation of a distributed data store in one embodiment where the identification space for the data objects may be partitioned by a hash function into several zones, such as Zone i 310, Zone j 312 Zone k 314, of distributed hash table 308. For each of these zones, there may be a corresponding system node with a computer storage medium, such as Node i 316, Node j 318, and Node k 320. The computer storage medium of each system node may generally include either index data 322 or object data 328 or both. For example, the computer storage medium of Node i 316 may only include object data, specifically object x 330. The index data 322 stored on a computer storage medium may include information about a replica set for each object such as an indication of system nodes where replicas of the object may be stored. For example, index data for object x 324 stored on Node j

Claim 7	Zhang
	318 may indicate that replicas of object x may be stored on Node i 316, Node j 318, and Node k 320. Each of these nodes may include a stored replica of object x such as a replica of object x 330 stored on Node i 316, a replica of object x 332 stored on Node j 318, and a replica of object x 336 stored on Node k 320." Zhang at [0046]. "In general, an index module on a node may hold the indices for data objects rooted at that node. The data module on a node may hold data objects and may also maintain reverse tables that may include references that point to the root nodes of those data objects stored. For instance, FIG. 3 shows dotted arrows from the replica of data object y 334 and the replica of data object y 338 pointing to index 326 for object y stored on Node k 320." Zhang at [0047].

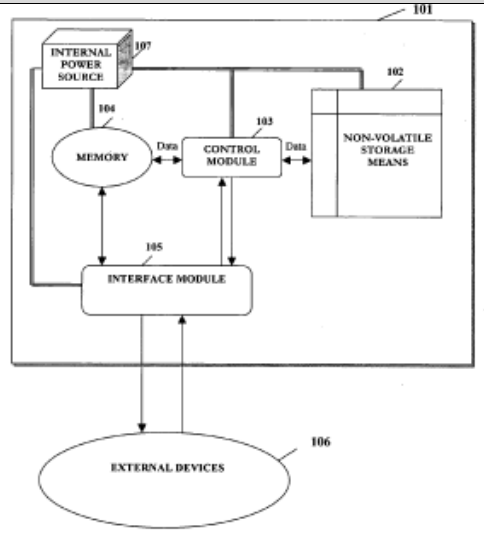
Claim 7	Zhang
	 *FIG. 3* Zhang at FIG. 3. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks

Claim 7	Zhang
	not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

Claim 7	Zhang
	 Figure 2: An example capsule disk. Sapuntzakis at 6. *See also* Sapuntzakis at 1-12. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally."

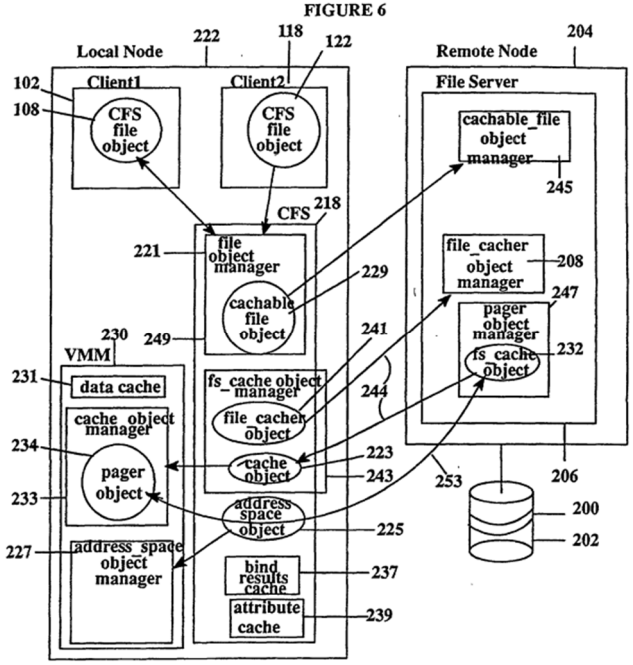
Claim 7	Zhang
	Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037]. "The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two

Claim 7	Zhang
	types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

Claim 7	Zhang
	 Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

Claim 7	Zhang
	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its

Claim 7	Zhang
	address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Zhang
	FIGURE 6  Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Zhang
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Zhang
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"Briefly, the present invention may provide a system and method for flexible object placement and soft-state indexing of objects in a distributed object store. A distributed object store may be provided by a large number of system nodes operably coupled to a network. A client executing an application may also be operably coupled to the network. A system node provided may include an access module for communicating with a client, an index module for building an index of a replicated data object, a data module for storing a data object on a computer readable medium, and a membership and routing module for detecting the

Claim 11	Zhang
	configuration of operable nodes in the distributed system. A system node may be configured as an index node that may have an index module but not a data module. A system node may also be configured as a data node that may have a data module but not an index module." Zhang at [0008]; *see also* [0025]-[0033]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure."

Claim 11	Zhang
	Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read

Claim 11	Zhang
	request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069].

Claim 11	Zhang
	See also Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-6

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 8,001,085 ("Kiselev")

Kiselev titled "Remote data access for local operations" to Oleg Kiselev was filed on November 25, 2003 issued August 16, 2011.[1] Kiselev is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Kiselev does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Kiselev or combine the teachings of Kiselev with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

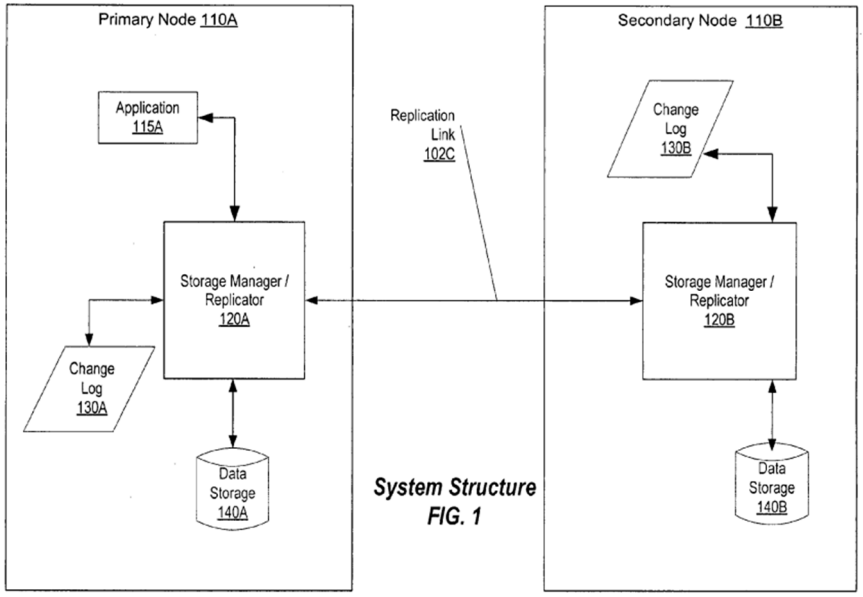
- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson"); and/or
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

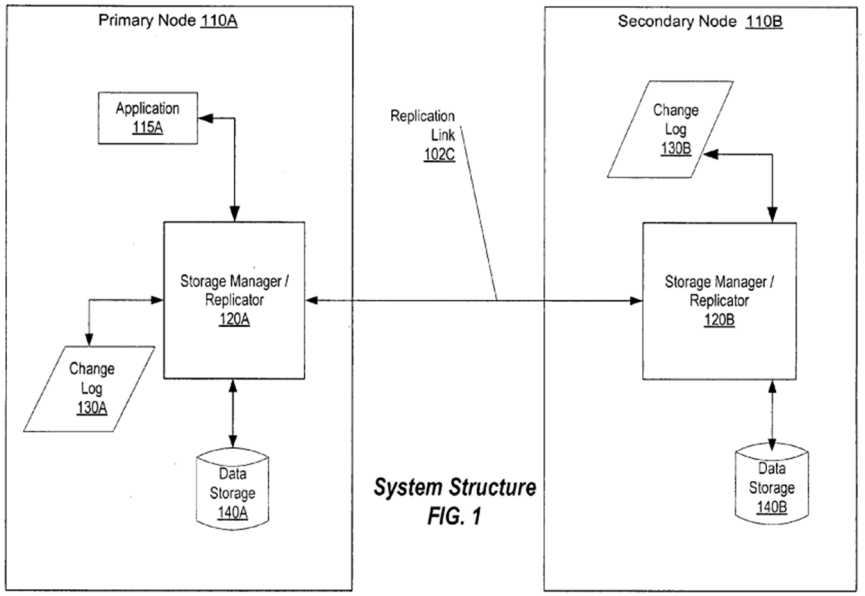
[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Kiselev
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Kiselev discloses: "A method, system, and computer-readable medium to maintain consistent, up-to-date copies of primary data. Primary data is maintained on a primary node, and point-in-time snapshots of the primary data are maintained on a secondary or remote node. Read operations can be directed to read some or all data from the remote node. When part of the data on the primary node becomes unavailable due to corruption, device failure, or communication failure, unavailable data can be read from the secondary node. When the determination is made that an unavailable portion of the data exists, a third data storage accessible by the first host can be created. Each subsequent change to the data in the first data storage can be written only to the third data storage. If an updated portion of the data is to be read, the updated portion can be read from the third data storage." Kiselev, Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"The present invention provides the ability to maintain consistent, up-to-date copies of primary data that are protected from corruption and that enable quick resumption of operations upon discovery of corruption of the primary data or failure of the primary node. Primary data is maintained on a primary node, and point-in-time snapshots of the primary data are maintained on a secondary or remote node. Read operations involve requesting some or all data to be read from a secondary host having access to the secondary data storage. When part of the data on the primary node becomes unavailable for reading due to corruption, device failure, or communication failure, the unavailable data can be requested from the secondary host. A previous version of the data that has become unavailable because the data have been overwritten on the primary data storage can also be requested from the secondary host. When the determination is made that some of the data is unavailable, a third data storage accessible by the primary host can be created. Each subsequent change to the data in the primary data storage can be written to the third data storage. If an updated portion of the data is to be read, the updated portion can be read from the third data storage."

Claim 7	Kiselev
	Kiselev, 3:21-29. "The present invention provides the ability to maintain consistent, up-to-date copies of primary data that are protected from corruption and that enable quick resumption of operations upon discovery of corruption of the primary data or failure of the primary node. Primary data is maintained on a primary node, and point-in-time snapshots are maintained on a secondary or remote node." Kiselev at 5:26-32.  Kiselev at FIG. 1.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously	"Data storage may take form as groups of one or more physical devices, such as one or more dynamic or static random access storage devices, one or more magnetic or optical data storage disks, or one or more other types of storage devices. With respect to backup copies of primary data, preferably the backup

Claim 7	Kiselev
contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	storage devices are direct access storage devices such as disks rather than sequential access storage devices such as tapes. Because disks are often grouped to form a logical storage volume that is used to store backup copies of primary data, the terms "storage" and "data storage" are used interchangeably herein with "storage volume;" however, one of skill in the art will recognize that the systems and processes described herein are also applicable to other types of storage and that the use of the term "storage volume" is not intended to be limiting. A storage volume storing the primary data is referred to herein as a primary volume, and a storage volume storing a backup copy of the primary data is referred to herein as a backup volume, secondary volume, or remote volume. A storage volume storing a snapshot of data is referred to herein as a snapshot volume." Kiselev at 4:64-5:16. "The present invention provides the ability to maintain consistent, up-to-date copies of primary data that are protected from corruption and that enable quick resumption of operations upon discovery of corruption of the primary data or failure of the primary node. Primary data is maintained on a primary node, and point-in-time snapshots are maintained on a secondary or remote node." Kiselev at 5:26-32. "The combination of the data from the primary and secondary nodes is referred to as a distributed overlay snapshot. A complete image of the data in the distributed overlay Snapshot may not be completely stored on either the primary or the secondary node, but rather can be constructed as a virtual image of the data from portions of the data stored on both nodes." Kiselev at 5:57-63. "FIG. 1 shows an example of a system environment in which the present invention may operate. Two nodes are shown, primary node 110A and secondary node 110B. Software programs application 115A and storage manager/replicator 120A operate on primary node 110A. Application 115A manages primary data that can be stored in change log 130A and data storage 140A.

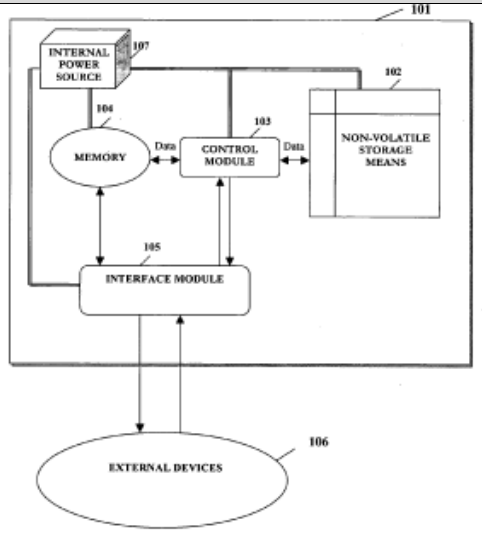
Claim 7	Kiselev
	Change log 130A can be considered to be a "staging area" to which changes to data are written before being written to data storage 140A. Change logs such as change log 130A, also referred to simply as logs, are known in the art and can be implemented in several different ways; for example, an entry in the log may represent an operation to be performed on a specified region of the data. Alternatively, the log may be structured to maintain a set of operations with respect to each region. Other types of log structures are also possible, and no particular type of implementation of change logs is required for operation of the invention. The invention can be practiced without using a log, although using a log is preferable in embodiments that use typical magnetic disk storage and cannot rely on availability of a large amount of non-volatile solid-state storage (NVRAM)." Kiselev at 6:5-27.  Kiselev at FIG. 1.

Claim 7	Kiselev
	"Storage manager/replicator 120A intercepts write operations to primary data by application 115A and replicates changes to the primary data to secondary node 110B. The type of replication performed by storage manager/replicator 120A can be synchronous, asynchronous, and/or periodic, as long as updates are applied consistently to both the primary and secondary data storage. While application 115A and storage manager/replicator 120A may run on the same computer system, such as primary node 110A, the hardware and software configuration represented by primary node 110A may vary. Application 115A and storage manager/replicator 120A may run on different computer systems, change log 130A may be stored in non-persistent or persistent data storage, and data storage 140A is a logical representation of a set of data stored on a logical storage device which may include one or more physical storage devices. Furthermore, while connections between application 115A, storage manager/replicator 120A, change log 130A, and data storage 140A are shown within primary node 110A, one of skill in the art will understand that these connections are for illustration purposes only and that other connection configurations are possible. For example, one or more of application 115A, storage manager/replicator 120A, change log 130A, and data storage 140A can be physically outside, but coupled to, the node represented by primary node 110A." Kiselev at 6:28-52. "Secondary data storage 140B is logically isolated from primary data storage 140A. Storage manager/replicator 120A of primary node 110A communicates over replication link 102C with storage manager/replicator 120B of secondary node 110B. Replication link 102C can be any known communication or networking link, including a shared device used for data transfer between hosts. Secondary node 110B also includes a change log 130B and data storage 140B for storing a replica of the primary data, and similar variations in hardware and software configuration of secondary node 110B are possible." Kiselev at 6:53-63.

Claim 7	Kiselev
	"These techniques include logging all changes to the known-to-be-stable copy of data without actually applying these changes to that stable copy and constructing a current version of the data from the log and the stable copy. Another technique includes using purely log-based storage such that changes to the data can be selectively applied when a current version of the data is constructed from the log. Yet another technique includes logging new data on the primary node in batches for periodic application to the remote copy and taking a snapshot of the remote copy before every batch update. The primary and secondary data storage may be configured the same or differently. For example, each data storage area may be configured to store data only in a log; in a log combined with a storage volume; or in a log, storage volume, and a set of one or more snapshots. Preferably, at least the secondary node is configured to maintain multiple versions of the data." Kiselev at 7:12-28. "Data logs. Another approach to enable recovery of previous versions of valid primary data involves logging data on the remote node. Data are written to a log when received at the secondary node and a large number of log updates are maintained in a cache memory. Prior to writing the data from the log into permanent data storage, a snapshot of the permanent data storage is written, and then a cached version of the log data is used to update the permanent data storage. This technique lowers the latency of persistently storing data on the remote node because sequential I/O latency when writing data from the log cache is usually 7-8 times lower than random I/O latency. Furthermore, the cost of maintaining point-in-time copies of the data (snapshots) is lowered because snapshots are taken relatively infrequently." Kiselev at 7:44-57. "In addition to the number of snapshots to be retained, a management decision must be made about the type(s) of snapshots to retain. One type of snapshot is a "full data" snapshot, in which a complete copy of every region of the corresponding storage volume is made. Another type of snapshot is a "copy-on-write" (COW)

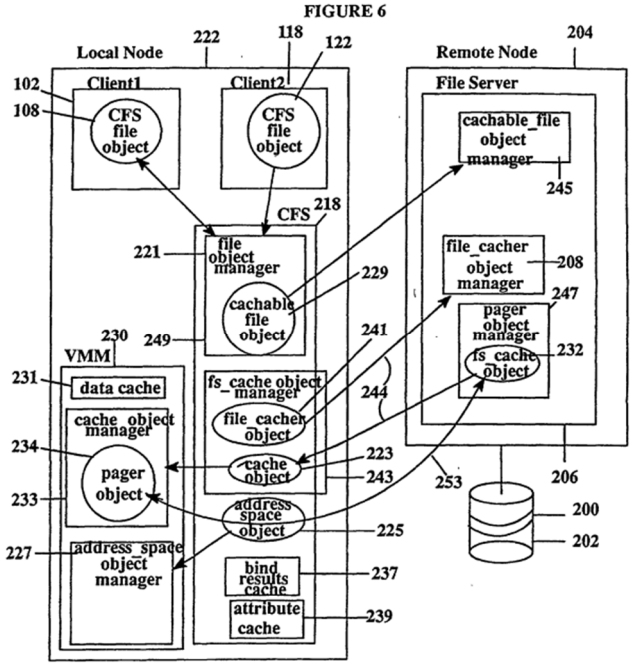
Claim 7	Kiselev
	snapshot. In a copy-on-write snapshot, before an update to a region of the primary data is made, the previous data in the region is copied to a copy-on-write snapshot. The copy-on-write snapshot typically contains only data for regions that change and is combined with the unchanged regions of the primary volume to provide a complete image of a previous state of the data. This scheme has an advantage that when the oldest snapshots need to be eliminated, no data must be moved from one location to another and the old copy-on-write snapshots can simply be deleted. One disadvantage is that the copy-on-write process is expensive because additional I/O operations are required for every write operation. Another type of snapshot is referred to as an "overlay" snapshot. Changes to data are written to an empty overlay storage object, and the previous version of the data remains unchanged in the main storage volume. In effect, because the main storage volume does not change, the main storage volume serves as a point-in-time snapshot of the data. This point-in-time snapshot of the main storage volume is combined with one or more overlay storage objects to create a virtual image of the data, which can be referred to as an "overlay snapshot." Note that an overlay snapshot does not physically exist in one data storage location, but rather is constructed as a virtual image when needed. This technique enables the integrity of an earlier image of the data to be preserved, while allowing fast write operations. Multiple overlay storage objects can be created to capture the state of the data in the main storage volume at different points in time. Overlay snapshots are further discussed with reference to FIGS. 2 and 3A through 3E." Kiselev at 8:59-9:27.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"Space-optimized snapshots are sometimes referred to as "cache-structured" snapshots because the data are organized for efficient use of space. In storing data in a cache-structured snapshot, a cache manager stores the data in a layout that efficiently stores data rather than using a strict logical-to-physical address translation. A mapping is used to track correspondence between the logical region of the data to the physical region of the underlying physical storage device. Upon responding to a read request, the cache manager uses the mapping identify the physical region of the storage device from which to read data for a given logical

Claim 7	Kiselev
	address, rather than performing a simple address translation to obtain the physical address." Kiselev at 10:18-30. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

Claim 7	Kiselev
	 Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

Claim 7	Kiselev
	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its

Claim 7	Kiselev
	address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Kiselev
	 Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Kiselev
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Kiselev
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"These techniques include logging all changes to the known-to-be-stable copy of data without actually applying these changes to that stable copy and constructing a current version of the data from the log and the stable copy. Another technique includes using purely log-based storage such that changes to the data can be selectively applied when a current version of the data is constructed from the log. Yet another technique includes logging new data on the primary node in batches for periodic application to the remote copy and taking a snapshot of the remote copy before every batch update.

Claim 11	Kiselev
	The primary and secondary data storage may be configured the same or differently. For example, each data storage area may be configured to store data only in a log; in a log combined with a storage volume; or in a log, storage volume, and a set of one or more snapshots. Preferably, at least the secondary node is configured to maintain multiple versions of the data." Kiselev at 7:12-28. "Data logs. Another approach to enable recovery of previous versions of valid primary data involves logging data on the remote node. Data are written to a log when received at the secondary node and a large number of log updates are maintained in a cache memory. Prior to writing the data from the log into permanent data storage, a snapshot of the permanent data storage is written, and then a cached version of the log data is used to update the permanent data storage. This technique lowers the latency of persistently storing data on the remote node because sequential I/O latency when writing data from the log cache is usually 7-8 times lower than random I/O latency. Furthermore, the cost of maintaining point-in-time copies of the data (snapshots) is lowered because snapshots are taken relatively infrequently." Kiselev at 7:44-57. "In addition to the number of snapshots to be retained, a management decision must be made about the type(s) of snapshots to retain. One type of snapshot is a "full data" snapshot, in which a complete copy of every region of the corresponding storage volume is made. Another type of snapshot is a "copy-on-write" (COW) snapshot. In a copy-on-write snapshot, before an update to a region of the primary data is made, the previous data in the region is copied to a copy-on-write snapshot. The copy-on-write snapshot typically contains only data for regions that change and is combined with the unchanged regions of the primary volume to provide a complete image of a previous state of the data. This scheme has an advantage that when the oldest snapshots need to be eliminated, no data must be moved from one location to another and the old copy-on-write snapshots can simply be deleted. One

Claim 11	Kiselev
	disadvantage is that the copy-on-write process is expensive because additional I/O operations are required for every write operation. Another type of snapshot is referred to as an "overlay" snapshot. Changes to data are written to an empty overlay storage object, and the previous version of the data remains unchanged in the main storage volume. In effect, because the main storage volume does not change, the main storage volume serves as a point-in-time snapshot of the data. This point-in-time snapshot of the main storage volume is combined with one or more overlay storage objects to create a virtual image of the data, which can be referred to as an "overlay snapshot." Note that an overlay snapshot does not physically exist in one data storage location, but rather is constructed as a virtual image when needed. This technique enables the integrity of an earlier image of the data to be preserved, while allowing fast write operations. Multiple overlay storage objects can be created to capture the state of the data in the main storage volume at different points in time. Overlay snapshots are further discussed with reference to FIGS. 2 and 3A through 3E." Kiselev at 8:59-9:27.

Exhibit 844-7

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2004/0172509 ("Takeda")

Takeda titled "Data processing system including storage systems" to Takahiko Takeda et al. was filed on June 23, 2003 and published on September 2, 2004.[1] Takeda is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Takeda does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Takeda or combine the teachings of Takeda with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson"); and/or
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Takeda
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Takeda discloses: "A data processing system includes a first storage system including a first host and a first storage subsystem. The first host has access to a first copy manager that is operable to manage a data replication operation. A second storage system includes a second host and a second storage subsystem. The second host has access to a second copy manager that is operable to manage a data replication operation. A first communication link is coupled to the first storage system and the second storage system to exchange management information between the first and second storage systems in order to manage the data replication operation. A data transfer path is configured to transfer data stored in the first storage subsystem to the second storage subsystem and replicate the data of the first storage subsystem in the second storage subsystem. The data transfer path is different from the first communication link." Takeda at Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"This data processing system comprises a first storage system 10 (hereinafter, "primary storage system" or "primary site") having a primary host 100A and a primary storage subsystem or disk array device 200A, a second storage system 20 (hereinafter, "secondary storage system" or "secondary site") having a secondary host 100B and a secondary storage subsystem or disk array device 200B, and a remote console 40. In the embodiments disclosed herein, disk array devices are used as the storage subsystems for illustration purposes. The storage subsystems may be storage devices other than disk array devices and are not limited thereto. The primary site and the primary disk array devices may be referred to as a first site and a first disk array device, respectively. Similarly, the secondary site and the secondary disk array devices may be referred to as a second site and a second disk array device, respectively." Takeda at [0035]. "Each host 100 (the primary host 100A and the secondary host 100B) is a computer that has a CPU 110, a main memory 120, and an input/output (I/O) processing

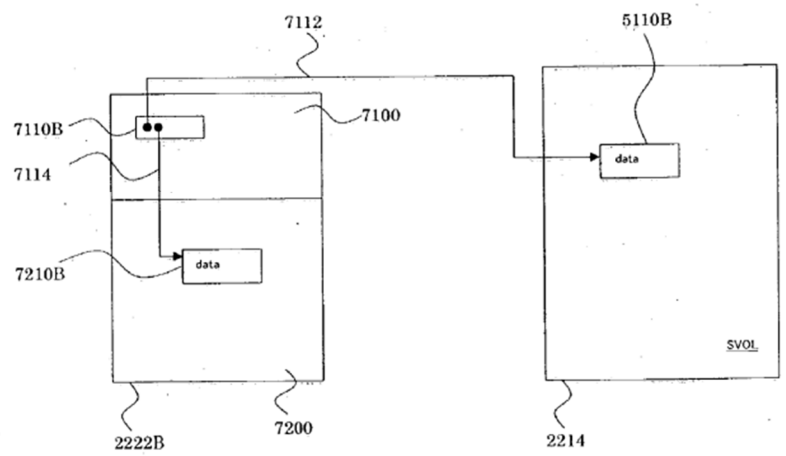
Claim 7	Takeda
	device 130. For example, it may be a workstation, a microcomputer, a mainframe computer, or the like." Takeda at [0036]. "Each disk array device 200 (the primary disk array device 200A and the secondary disk array device 200B) has a storage controller 210, a plurality of disk units 220, and an SVP (Service Processor) 230. The storage controller 210 has a host adapter 211, cache memory 212, a disk adapter 213, a processor 214, and control memory 215." Takeda at [0037]. "The primary host 100A is connected with the primary disk array device 200A and the secondary host 100B is connected with the secondary disk array device 200B, respectively, with a Fibre Channel 66. The CPU 110 and the main memory 120 of each host 100 are connected to the host adapter 211 of the disk array device 200 through the I/O processing device 130 and the Fibre Channel 66." Takeda at [0038].
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"This data processing system comprises a first storage system 10 (hereinafter, "primary storage system" or "primary site") having a primary host 100A and a primary storage subsystem or disk array device 200A, a second storage system 20 (hereinafter, "secondary storage system" or "secondary site") having a secondary host 100B and a secondary storage subsystem or disk array device 200B, and a remote console 40. In the embodiments disclosed herein, disk array devices are used as the storage subsystems for illustration purposes. The storage subsystems may be storage devices other than disk array devices and are not limited thereto. The primary site and the primary disk array devices may be referred to as a first site and a first disk array device, respectively. Similarly, the secondary site and the secondary disk array devices may be referred to as a second site and a second disk array device, respectively." Takeda at [0035].

Claim 7	Takeda
	"Each host 100 (the primary host 100A and the secondary host 100B) is a computer that has a CPU 110, a main memory 120, and an input/output (I/O) processing device 130. For example, it may be a workstation, a microcomputer, a mainframe computer, or the like." Takeda at [0036]. "Each disk array device 200 (the primary disk array device 200A and the secondary disk array device 200B) has a storage controller 210, a plurality of disk units 220, and an SVP (Service Processor) 230. The storage controller 210 has a host adapter 211, cache memory 212, a disk adapter 213, a processor 214, and control memory 215." Takeda at [0037]. "The primary host 100A is connected with the primary disk array device 200A and the secondary host 100B is connected with the secondary disk array device 200B, respectively, with a Fibre Channel 66. The CPU 110 and the main memory 120 of each host 100 are connected to the host adapter 211 of the disk array device 200 through the I/O processing device 130 and the Fibre Channel 66." Takeda at [0038]. "On a storage controller 210 of each disk array device 200, a copy program 2110 and a journal-backup/restore program 2120 are executed by the processor 214. These programs are stored in control memory 215. The journal-backup/restore program 2120 is composed of a journal backup program and a journal restore program. Moreover, the storage controller 210 handles I/O requests to the disk device 220 based on instructions from the hosts in addition to the copy program 2110 and the journal-backup/restore program 2120." Takeda at [0044]. "At step 9100, by using a graphical user interface ("GUI"), which is included in either the host 100 or the remote console 40, the user inputs a pair generation command to the host 100 (either of the primary host 100A or the secondary host

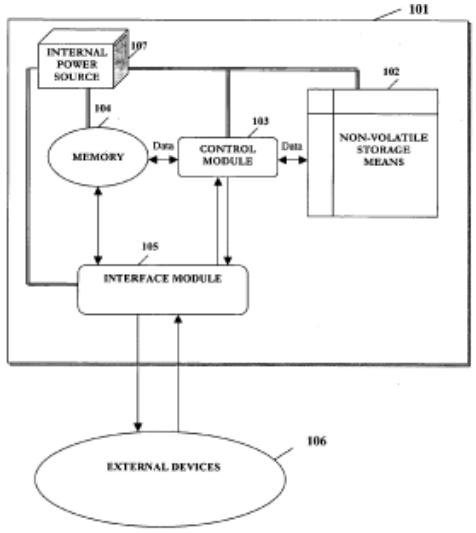
Claim 7	Takeda
	100B may be used). The pair generation command is a command that associates a first volume (hereinafter, "PVOL") 2212 of the primary disk array device 200A, which is targeted for journal acquisition, and a second volume (hereinafter, "SVOL") 2214 of the secondary disk array device 200B, which is a replication pair of PVOL." Takeda at [0050]. "Each disk array device 200 notifies information on the logical volumes registered in the journal volume pool, for example, physical addresses in the disk array device 200A, the storage capacity of the physical volume, etc. to the respective host 100. The host 100 that received the pair generation command selects an appropriate volume as the journal volume based on the information notified by the disk array device. In one implementation, it does not matter whether the selected volume is a single volume or a volume group, i.e., a plurality of volumes. If it is a plurality of volumes, the volumes are treated as a single virtual volume." Takeda at [0055]. "The user can specify whether or not a normal application executed on the host 100 is capable of issuing I/O request to the journal volume. There are two cases: (1) one is that a volume allocated on the disk device 220 used for normal I/O processing is selected as the journal volume; and (2) the other is that a volume that the host 100 cannot use for the normal I/O processing is selected as the journal volume." Takeda at [0056]. "In the former case, the journal can be seen from a normal application on the host 100 or from another host connected to the disk array device 200 through the Fibre Channel. Therefore, acquisition and control of statistics on the journal can be performed by a normal application, but there is possibility of destroying the journal erroneously." Takeda at [0057].

Claim 7	Takeda
	"In the latter case, the journal of the host 100 is allowed to be seen only when the host 100 executing the asynchronous copy manager 150 makes each disk array device 200 execute the journal restore program or the copy program. Therefore, the journal is not destroyed by the normal I/O processing performed by the host 100." Takeda at [0058]. "Thereafter, a journal processing is performed, e.g., in the first storage system 10 (Step 9120). The journal processing includes an acquisition operation 9122, a copy operation 9124, and a restoration operation 9126. The journal acquisition operation or process 9122 involves the primary disk array device 200A acquiring the journal for PVOL 2212 after receiving a journal acquisition start command from the primary host 100. After receiving the command, the primary disk array device. 200A commence storing journal data and metadata in the journal volume 2222A (Step 9220) after each write operation to PVOL 2212 (Steps 9200 and 9210). The journal data is a copy of the data written or updated according to the write command. The metadata provides information relating to a time when the update data is stored in the PVOL 2212, the storing address of the written (updated) data, address of the corresponding journal data in the journal data area, and the length of the data. A journal is composed of the journal data and corresponding metadata." Takeda at [0059]. "Generally, each of PVOL, SVOL and the journal volumes is managed in units of a predetermined logical block, e.g., 512 KB. Each of the logical blocks is given a logical block address (hereinafter, "LBA")." Takeda at [0083]. "The primary journal volume has a metadata area 7100 and a journal data area 7200. In the journal data area 7200, the previously described journal data 7210, i.e., the copy of data 5110 stored in PVOL by the write command, is stored. In the metadata area 7100, the previously described metadata 7110, i.e., information indicating a time when each update data is stored in PVOL 2212, the storing address 7112 of the written (updated) data, address 7114 of the corresponding

Claim 7	Takeda
	journal data 7210 in the journal data area 7200, and the length of the data, are stored." Takeda at [0084]. "FIG. 17 shows correspondence of a secondary journal volume 2222B and SVOL 2214 used in this embodiment. The secondary journal volume also has a metadata area 7100 and a journal data area 7200. In the metadata area 7100, the metadata 7110B, which is copied from the metadata area of the primary journal volume 2222A, is stored. In the journal data area 7200, the journal data 7210B, which corresponds to the metadata 7110B and copied from the journal data area of the primary journal volume, is stored." Takeda at [0087]. "While the metadata is the information about the update of PVOL 2212, its address information 7114 shows the address of the corresponding journal data 7210, which is copied to the journal data area in the secondary journal volume. Further, by copying journal data 7210 from the journal data area 7200 of the secondary journal volume 2222B to the address in SVOL 2214 corresponding to the address 7112, the update of PVOL 2212 can be reflected in SVOL 2214." Takeda at [0088]. "FIG. 5 shows the journal data areas of the primary journal volume and of the secondary journal volume according to the present embodiment." Takeda at [0089]. "The primary journal volume and the secondary journal volume are addressed with LBAs, and each LBA is brought into correspondence in a 1-to-1 manner." Takeda at [0090]. "The journal data area 7200 that the primary journal volume has is differentiated into a journal-stored area 2232, 2233, and 2234, in which the journal data are

Claim 7	Takeda
	stored, and an already-purged area 2231 in which no journal data is stored. The already-purged area is used for storing new journal data of PVOL 2212." Takeda at [0091]. Fig. 17  Takeda at FIG. 17. "The primary disk array device 200A receives the journal copy request command and transmits the specified journal to the secondary disk array device 200B by issuing a write command to the secondary disk array device 200B (Step 5600). The secondary disk array device 200B stores the journal received from the first device 200A in an area of the secondary journal volume specified by the command." Takeda at [0133]. "Subsequently, the secondary host 100B, using an asynchronous copy manager 150, reads the journal from the secondary journal volume and restores data to SVOL 2214 (Step 5900)." Takeda at [0134].

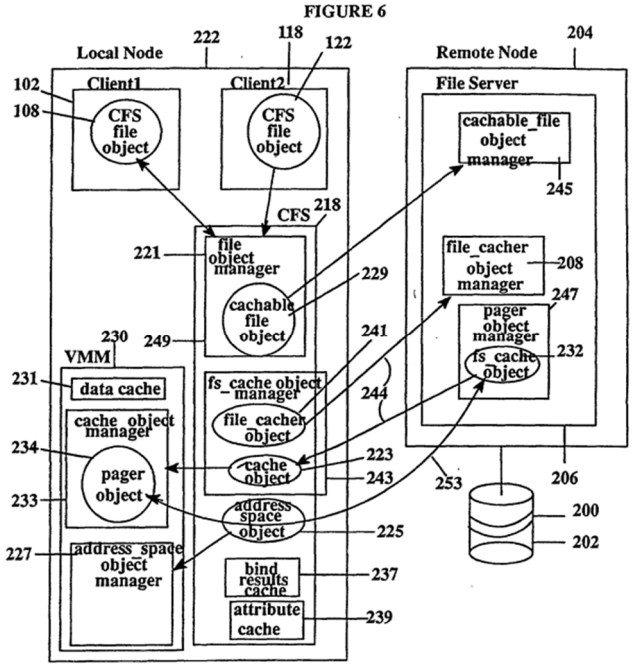
Claim 7	Takeda
	"The secondary host 100B performs pointer management of the secondary journal volume, and notifies information needed to make a journal copy request 5450 (for example, the already-restored LBA 4241 that is needed to calculate the address for copy) to the primary host 100A." Takeda at [0135]. "In this embodiment, by the instruction of the primary host 100A, the primary disk array device 200A issues the write command that requires the secondary disk array device 200B to write the initial data of PVOL 2212 into SVOL 2214 of the secondary disk array device 200B sequentially, whereby the initial copy is achieved." Takeda at [0136]. "After all initial data has been written in the secondary disk array device 200B from PVOL, the primary disk array device 200A executing the copy program 2110 reports completion of the initial copy to the primary host 100A. The primary host 100A receives this report. After that, SVOL 2214 restored in the secondary site 20 can be treated as a volume reflecting the contents of PVOL 2212." Takeda at [0137].
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"Each disk array device 200 (the primary disk array device 200A and the secondary disk array device 200B) has a storage controller 210, a plurality of disk units 220, and an SVP (Service Processor) 230. The storage controller 210 has a host adapter 211, cache memory 212, a disk adapter 213, a processor 214, and control memory 215." Takeda at [0037]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other

Claim 7	Takeda
	references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].  Fig. 1

Claim 7	Takeda
	Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module." Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file

Claim 7	Takeda
	server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches."

Claim 7	Takeda
	Nelson at [0040].



Nelson at FIG. 6.

See also Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4.

> "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file

Claim 7	Takeda
	system in persistent storage. Status includes, for example, information about what files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Takeda
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"Each disk array device 200 notifies information on the logical volumes registered in the journal volume pool, for example, physical addresses in the disk array device 200A, the storage capacity of the physical volume, etc. to the respective host 100. The host 100 that received the pair generation command selects an appropriate volume as the journal volume based on the information notified by the disk array device. In one implementation, it does not matter whether the selected volume is a

Claim 11	Takeda
	single volume or a volume group, i.e., a plurality of volumes. If it is a plurality of volumes, the volumes are treated as a single virtual volume." Takeda at [0055]. "After receiving a journal restore request command from the secondary host 100B (Step 9400), the secondary disk array device 200B restores the data stored in the SVOL 2214 by using the journal stored in the journal volume 2222B (Step 9410). This process is referred to as a journal restore process. The details of journal restore process will be described later." Takeda at [0068]. "The secondary disk array device 200B performs the journal restore process 9126 by executing a journal restore program 2124 in the journal-backup/restore program 2120. The journal restore program 2124 restores journal in the journal volume 2222B, so that the data volume 2214 reflects the updated data image of PVOL 2212." Takeda at [0070].

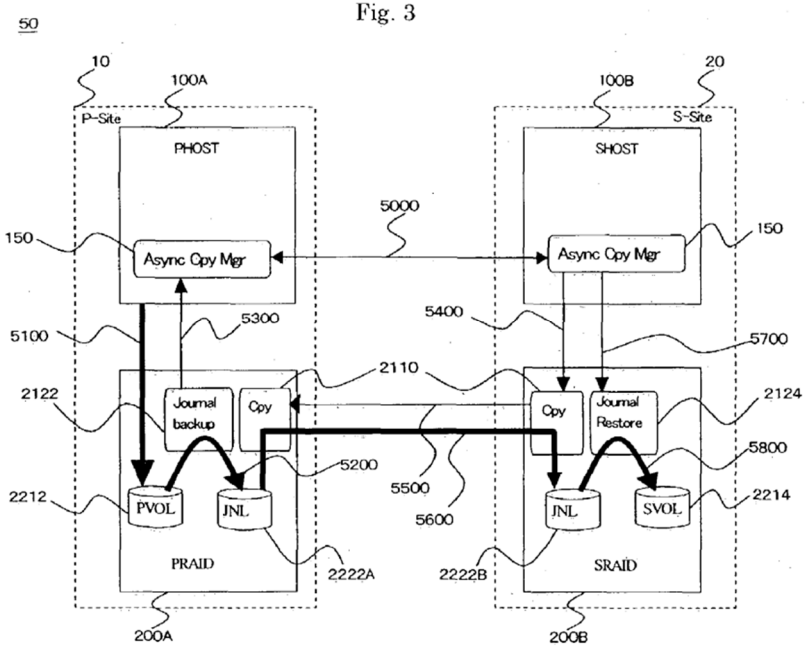
Claim 11	Takeda
	Fig. 3  Takeda at FIG. 3.

Exhibit 844-8

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 6,671,782 ("Menon")

Menon titled "Method and apparatus for processing read requests in a shared disk system" to Jaishankar Moothedath Menon was filed on March 29, 2000 issued on December 30, 2003.[1] Menon is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Menon does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Menon or combine the teachings of Menon with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

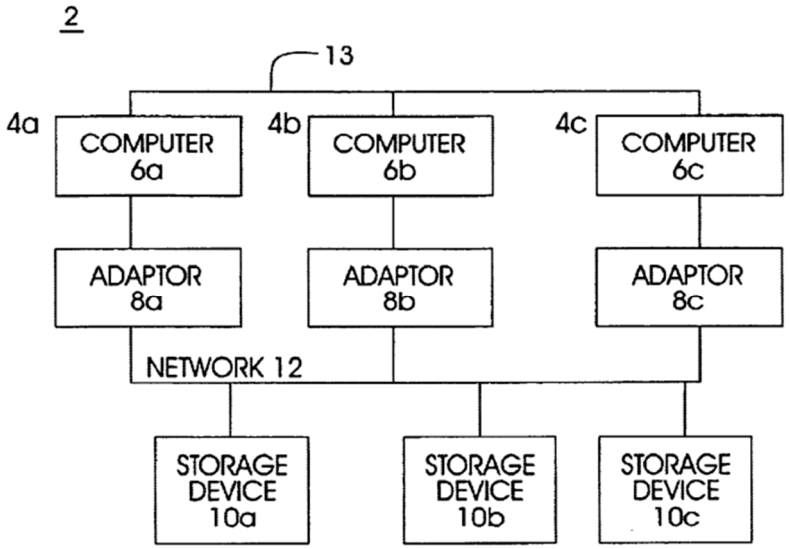
As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

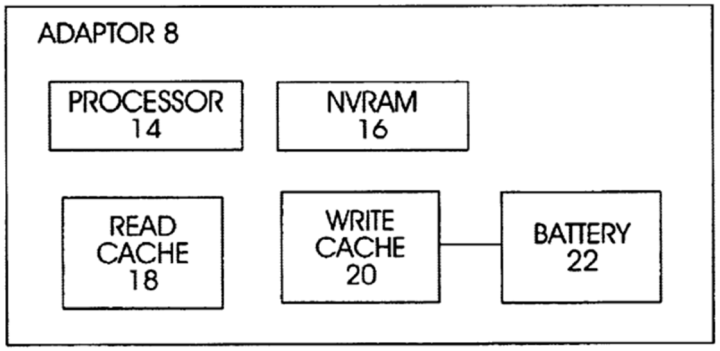
pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

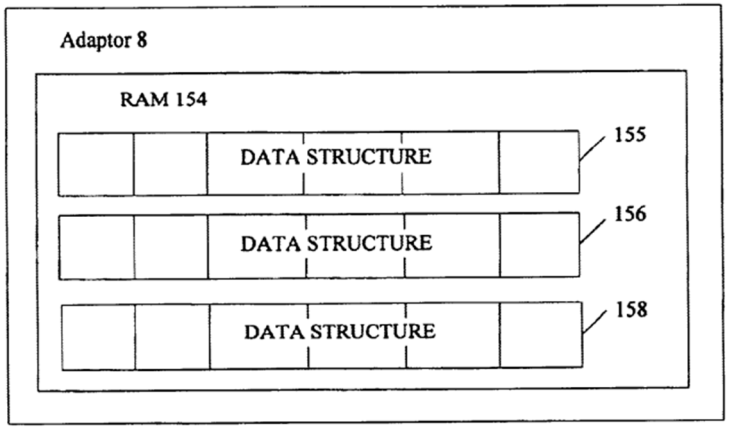
Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Menon
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Menon discloses: "A shared device environment having multiple nodes (4 a -4 c), each node containing a computer (6 a -6 c), an adaptor (8 a -8 c) and multiple storage devices (10 a -10 c). Adaptors (8 a -8 c) facilitate read access to storage devices (10 a -10 c) such that the data records accessed are the latest version data records. Adaptors (8 a -8 c) are operative to provide the latest version of data blocks requested by computers (6 a -6 c) by first searching the read cache of each adaptor for data blocks requested by computers (6 a -6 c). If the requested data blocks are not found in the read cache of any adaptors, then data structures (155, 156 and 158) within the adaptors are searched for any data blocks waiting to be added to the read cache of any adaptors. If no data blocks are found, then the read access is conducted directly from the storage device itself." Menon at Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"FIG. 1 illustrates a shared device environment comprised of nodes 4 a, b, c. Each node includes a computer 6 a, b, c, i.e., central processing unit, and an adaptor card 8 a, b, c. A plurality of storage devices 10 a, b, c interface via a network 12 to the adaptor cards 8 a, b, c and attached computers 6 a, b, c. The computer 6 a, b, c may be a personal computer, workstation, mainframe, etc. The adaptor cards 8 a, b, c interface with their respective computers 6 a, b, c via a PC bus, such as the PCI bus, and include one or more disk interface ports, such as SCSI or Serial Storage Architecture (SSA) ports. The adaptor cards 8 a, b, c include logic to execute the RAID algorithms. The storage devices 10 a, b, c may be any suitable non-volatile storage device known in the art, including hard disk drives, magnetic tape, optical disks, non-volatile RAM, holographic units, etc. The nodes 4 a, b, c and storage devices 10 a, b, c interface via the network 12, which is preferably a high speed interconnect, such as SCSI, SSA, SNA, SAN, FDDI, etc. Additionally, the network 12 may be a SCSI or SSA bus. In further embodiments more nodes than shown may be included in the shared device system 2. Each node may include multiple adaptors, multiple processors and/or local (non-shared) storage devices."

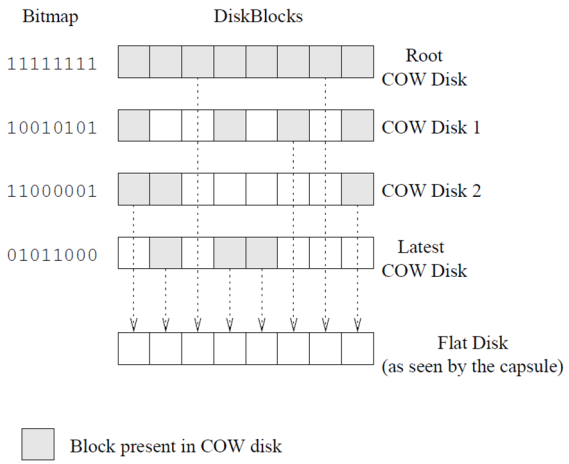
Claim 7	Menon
	Menon at 3:55-4:10.  *FIG. 1* Menon at FIG. 1. "FIG. 2 illustrates a preferred embodiment of the adaptors 8 a, b, c. Each adaptor 8 a, b, c includes a processor 14 a, b, c, a non-volatile RAM 16 a, b, c for storing control information, a read cache 18 a, b, c, and a write cache 20 a, b, c. The read 18 a, b, c and write 20 a, b, c caches may be comprised of volatile memory, such as RAM, or a non-volatile memory unit, e.g., non-volatile RAM. In certain embodiments, the read cache 18 a, b, c and write cache 20 a, b, c may be areas within the same memory device or located within separate memory devices. In further embodiments, there may be no read 18 a, b, c and/or nvrite 20 a, b, c caches. In preferred embodiments, the write caches 20 a, b, c contain dirty blocks, which is data intended for a block in the storage device 10 a, b, c that is more recent than the block actually maintained in the storage device 10 a, b, c. Once the data is written

Claim 7	Menon
	from the write cache 20 a, b, c to the storage device 10 a, b, c, the copy of the data in the cache is "clean." Because the write cache 20 a, b, c only maintains "dirty" blocks, the clean copy in the cache after the update is considered to be in the read cache 18 a, b, c, not the write cache 10 a, b, c anymore. The components of the adaptors 8 a, b, c may be implemented as PC cards such the PC ServeRAID SCSI adaptor from IBM. Alternatively, components and functionality of the adaptors 8 a, b, c could be implemented in the computers 6 a, b, c." Menon at 4:37-61.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"FIG. 2 illustrates a preferred embodiment of the adaptors 8 a, b, c. Each adaptor 8 a, b, c includes a processor 14 a, b, c, a non-volatile RAM 16 a, b, c for storing control information, a read cache 18 a, b, c, and a write cache 20 a, b, c. The read 18 a, b, c and write 20 a, b, c caches may be comprised of volatile memory, such as RAM, or a non-volatile memory unit, e.g., non-volatile RAM. In certain embodiments, the read cache 18 a, b, c and write cache 20 a, b, c may be areas within the same memory device or located within separate memory devices. In further embodiments, there may be no read 18 a, b, c and/or nvrite 20 a, b, c caches. In preferred embodiments, the write caches 20 a, b, c contain dirty blocks, which is data intended for a block in the storage device 10 a, b, c that is more recent than the block actually maintained in the storage device 10 a, b, c. Once the data is written from the write cache 20 a, b, c to the storage device 10 a, b, c, the copy of the data in the cache is "clean." Because the write cache 20 a, b, c only maintains "dirty" blocks, the clean copy in the cache after the update is considered to be in the read cache 18 a, b, c, not the write cache 10 a, b, c anymore. The components of the adaptors 8 a, b, c may be implemented as PC cards such the PC ServeRAID SCSI adaptor from IBM. Alternatively, components and functionality of the adaptors 8 a, b, c could be implemented in the computers 6 a, b, c." Menon at 4:37-61.

Claim 7	Menon
	 *FIG. 2* Menon at FIG. 2. "FIGS. 4 and 5 illustrate logic implemented in the adaptors 8 a, b, c, to update parity Pi when simultaneously updating different data blocks Di in the same parity group. The logic of FIGS. 4 and 5 utilizes the NVRAM 16 to store partial parity data that keeps track of changes that must be made to the parity groups as a result of updating data. With respect to FIG. 4, control begins at block 30 which represents an adaptor, e.g., updating adaptor 8 a, receiving a request to update a block of data from D to D′ on a disk, e.g., block B in disk i. Disk i may be one of the storage devices 10 a, b, c or a storage subunit within one of the storage devices 10 a, b, c. Control transfers to block 32 which represents the updating adaptor 8 a reading the old data D from block B in disk i via the network 12. Control then transfers to block 34 which represents the updating adaptor 8 a calculating partial parity for the data update from the update D′ and old data D. In preferred embodiments, partial parity is calculated as D xor D′. However, in alternative embodiments, alternative means known in the art for calculating parity may be used." Menon at 6:11-38. "In the embodiments utilizing read caches 18 a, b, c, the adaptors 8 a, b, c maintain a data structure indicating data cached at other remote adaptors. Preferred

Claim 7	Menon
	embodiments are described with respect to a two adaptor 8 a, b system. However, the logic could be extended to an N adaptor case. FIG. 8 illustrates a RAM 154 a, b, c within each adaptor 8 a, b, c storing three data structures. The first data structure 155 is a list of data blocks waiting to be added to the read cache 18 a, b, c. Until the adaptor 8 a, b, c adds the data blocks to the read cache 18 a, b, c they are maintained in the adaptors RAM 154 a, b, c. Data structure 156 a, b, c is a list of blocks recently updated by the adaptor 8 a, b, c and data structure 158 a, b, c is a list of blocks in the other adapter's read cache 18 a, b, c, i.e., a directory of the other adaptor's read cache. Each adaptor 8 a, b, c also maintains a list of inconsistency groups in its NVRAM 16 a, b, c." Menon at 11:1-16.  FIG. 8 Menon at FIG. 8. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and

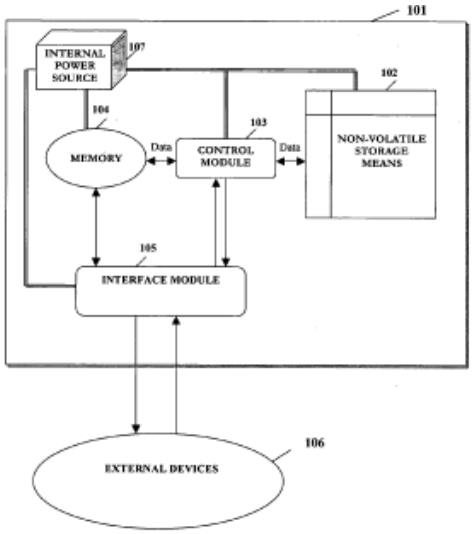
Claim 7	Menon
	Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

Claim 7	Menon
	 Figure 2: An example capsule disk. Sapuntzakis at 6. *See also* Sapuntzakis at 1-12. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write

Claim 7	Menon
	requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037].

Claim 7	Menon
	"The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"FIG. 2 illustrates a preferred embodiment of the adaptors 8 a, b, c. Each adaptor 8 a, b, c includes a processor 14 a, b, c, a non-volatile RAM 16 a, b, c for storing control information, a read cache 18 a, b, c, and a write cache 20 a, b, c. The read 18 a, b, c and write 20 a, b, c caches may be comprised of volatile memory, such as RAM, or a non-volatile memory unit, e.g., non-volatile RAM. In certain embodiments, the read cache 18 a, b, c and write cache 20 a, b, c may be areas within the same memory device or located within separate memory devices. In further embodiments, there may be no read 18 a, b, c and/or nvrite 20 a, b, c caches. In preferred embodiments, the write caches 20 a, b, c contain dirty blocks, which is data intended for a block in the storage device 10 a, b, c that is more recent than the block actually maintained in the storage device 10 a, b, c. Once the data is written from the write cache 20 a, b, c to the storage device 10 a, b, c, the copy of the data in the cache is "clean." Because the write cache 20 a, b, c only maintains "dirty" blocks, the clean copy in the cache after the update is considered to be in the read cache 18 a, b, c, not the write cache 10 a, b, c anymore. The components of the adaptors 8 a, b, c may be implemented as PC cards such the PC ServeRAID SCSI adaptor from IBM. Alternatively, components and functionality of the adaptors 8 a, b, c could be implemented in the computers 6 a, b, c." Menon at 4:37-61.

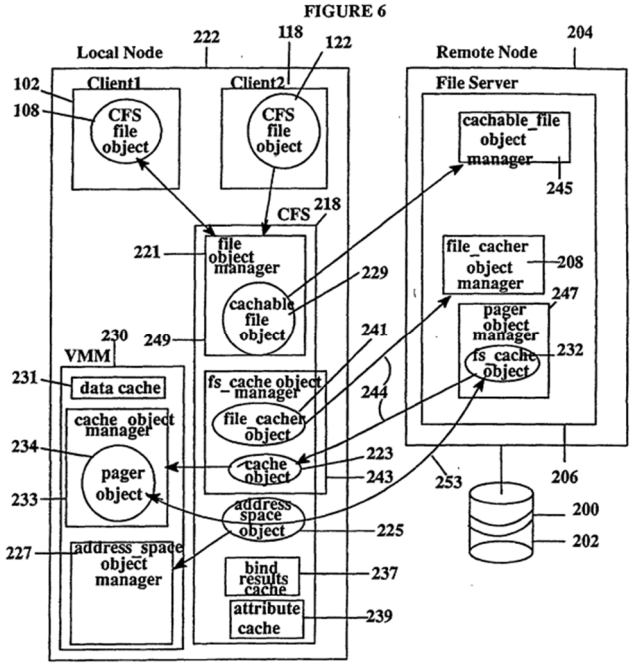
Claim 7	Menon
	"In the embodiments utilizing read caches 18 a, b, c, the adaptors 8 a, b, c maintain a data structure indicating data cached at other remote adaptors. Preferred embodiments are described with respect to a two adaptor 8 a, b system. However, the logic could be extended to an N adaptor case. FIG. 8 illustrates a RAM 154 a, b, c within each adaptor 8 a, b, c storing three data structures. The first data structure 155 is a list of data blocks waiting to be added to the read cache 18 a, b, c. Until the adaptor 8 a, b, c adds the data blocks to the read cache 18 a, b, c they are maintained in the adaptors RAM 154 a, b, c. Data structure 156 a, b, c is a list of blocks recently updated by the adaptor 8 a, b, c and data structure 158 a, b, c is a list of blocks in the other adapter's read cache 18 a, b, c, i.e., a directory of the other adaptor's read cache. Each adaptor 8 a, b, c also maintains a list of inconsistency groups in its NVRAM 16 a, b, c." Menon at 11:1-16. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple

Claim 7	Menon
	servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].  Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory

Claim 7	Menon
	module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module." Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the

Claim 7	Menon
	existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Menon



Nelson at FIG. 6.

See also Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4.

> "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Menon
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Menon
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"FIGS. 4 and 5 illustrate logic implemented in the adaptors 8 a, b, c, to update parity Pi when simultaneously updating different data blocks Di in the same parity group. The logic of FIGS. 4 and 5 utilizes the NVRAM 16 to store partial parity data that keeps track of changes that must be made to the parity groups as a result of updating data. With respect to FIG. 4, control begins at block 30 which represents an adaptor, e.g., updating adaptor 8 a, receiving a request to update a block of data from D to D′ on a disk, e.g., block B in disk i. Disk i may be one of the storage devices 10 a, b, c or a storage subunit within one of the storage devices

Claim 11	Menon
	10 a, b, c. Control transfers to block 32 which represents the updating adaptor 8 a reading the old data D from block B in disk i via the network 12. Control then transfers to block 34 which represents the updating adaptor 8 a calculating partial parity for the data update from the update D′ and old data D. In preferred embodiments, partial parity is calculated as D xor D′. However, in alternative embodiments, alternative means known in the art for calculating parity may be used." Menon at 6:11-38. "In the embodiments utilizing read caches 18 a, b, c, the adaptors 8 a, b, c maintain a data structure indicating data cached at other remote adaptors. Preferred embodiments are described with respect to a two adaptor 8 a, b system. However, the logic could be extended to an N adaptor case. FIG. 8 illustrates a RAM 154 a, b, c within each adaptor 8 a, b, c storing three data structures. The first data structure 155 is a list of data blocks waiting to be added to the read cache 18 a, b, c. Until the adaptor 8 a, b, c adds the data blocks to the read cache 18 a, b, c they are maintained in the adaptors RAM 154 a, b, c. Data structure 156 a, b, c is a list of blocks recently updated by the adaptor 8 a, b, c and data structure 158 a, b, c is a list of blocks in the other adapter's read cache 18 a, b, c, i.e., a directory of the other adaptor's read cache. Each adaptor 8 a, b, c also maintains a list of inconsistency groups in its NVRAM 16 a, b, c." Menon at 11:1-16. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or

Claim 11	Menon
	Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop

Claim 11	Menon
	preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory."

Claim 11	Menon
	Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-9

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2003/0236850 ("Kodama")

Kodama titled "Storage system for content distribution" to Shoji Kodama was filed on March 21, 2002 and published on December 25, 2003.[1] Kodama is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Kodama does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Kodama or combine the teachings of Kodama with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to

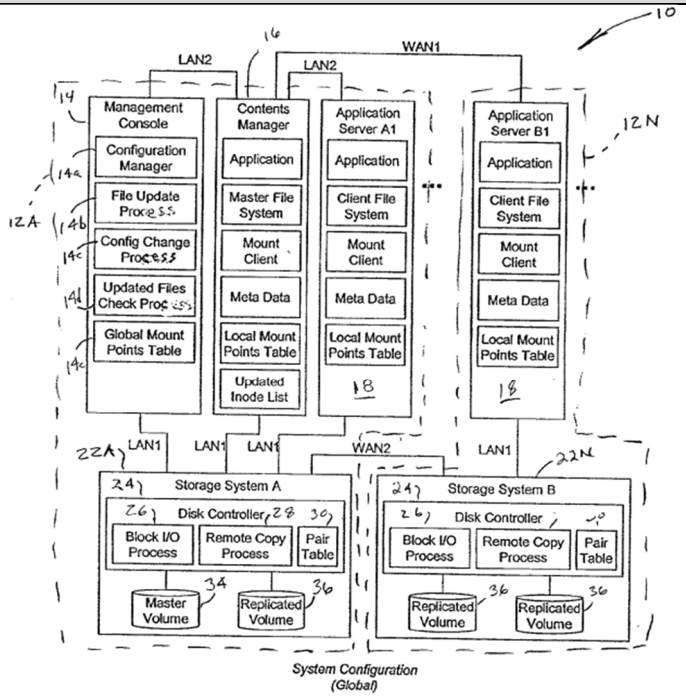
[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

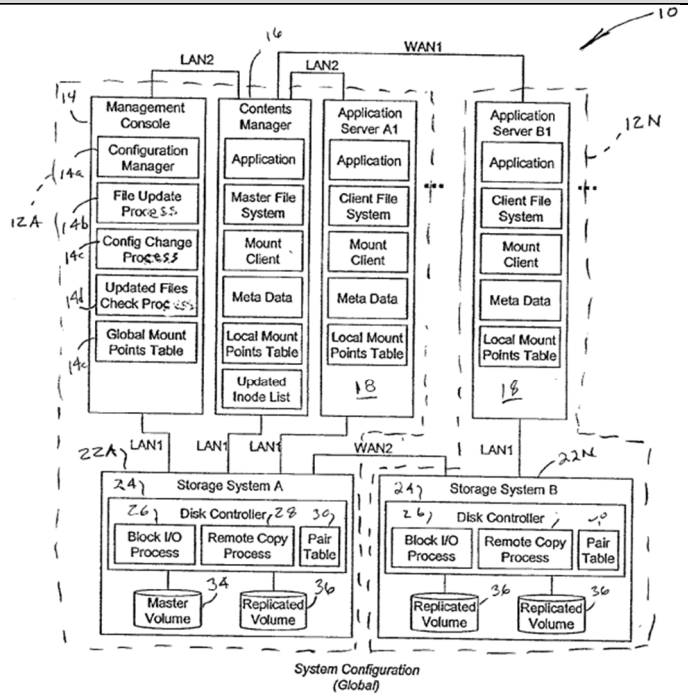
Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Kodama
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Kodama discloses: "A processing system comprises a primary storage system and a plurality of secondary storage systems. The primary storage system is managed by a contents manager that keeps a file system used to make file level I/O accesses to a controller that translates the file access to block I/O to access the magnetic media for the data. The secondary storage systems are available to and accessed by application servers, also using file level I/O. The content manager may change a file or data on the primary storage system and, if so, will also change the file system accordingly. The file system is then made available to the application servers, after the controller transfers changed files to the secondary storage systems using block level I/O." Kodama at Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"This invention is directed to a processing system having multiple copies of data distributed from a primary or master storage system to a number of secondary storage systems. The invention provides a system and a method that allows applications executing on servers to make file-level I/O accesses to the storage systems, yet permit data transfers between the storage systems to use block-level I/O." Kodama at [0007]. "In accordance with the invention, a primary or master data center includes a primary processing facility and a storage system whereat is maintained a master copy of data. Stored with the data are data structures (directory lists, inode lists, and the like) that identify the location of the data. Periodically, changes to the data (deletions, modifications, etc.) are made to the data. The changes are written to a previously unused area of the storage media containing the data. Changes to the data are followed by modifications to the data structures to identify the location(s) of the changed data. Then, the changed data and data structure are placed on a remote copy queue. A remote copy process periodically scans the remote copy queue for any changes made since the last scan, and if changed data is found, the

Claim 7	Kodama
	changed data, together with the associated modified data structure, will be copied to one or more secondary storage systems and made available to application servers coupled to the secondary storage systems." Kodama at [0050]. "Turning now to the Figures, FIG. 1 is an illustration of a basic data processing system, designated generally with the reference numeral 10, employing the present invention. As FIG. 1 shows, the data processing system 10 may include a number of data centers, here illustrated by data centers 12A and 12N. The primary data center 12A includes a management console 14, a contents manager 16, and may or may not include an application server 18 and a storage system 22A. The other data centers 12, such as data center 12N, are application servers, each including a storage system 22N." Kodama at [0053].

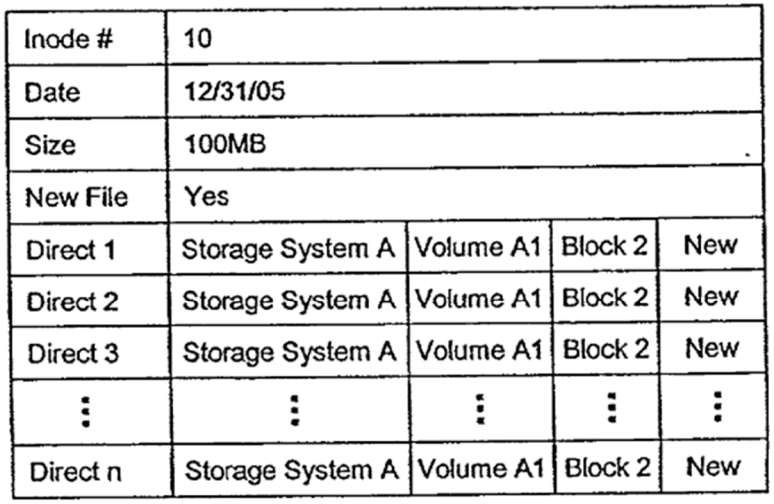
Claim 7	Kodama
	 FIG_1 Kodama at FIG. 1.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image	"This invention is directed to a processing system having multiple copies of data distributed from a primary or master storage system to a number of secondary storage systems. The invention provides a system and a method that allows applications executing on servers to make file-level I/O accesses to the storage systems, yet permit data transfers between the storage systems to use block-level I/O." Kodama at [0007]. "In accordance with the invention, a primary or master data center includes a primary processing facility and a storage system whereat is maintained a master

Claim 7	Kodama
that are unchanged by respective compute nodes; and	copy of data. Stored with the data are data structures (directory lists, inode lists, and the like) that identify the location of the data. Periodically, changes to the data (deletions, modifications, etc.) are made to the data. The changes are written to a previously unused area of the storage media containing the data. Changes to the data are followed by modifications to the data structures to identify the location(s) of the changed data. Then, the changed data and data structure are placed on a remote copy queue. A remote copy process periodically scans the remote copy queue for any changes made since the last scan, and if changed data is found, the changed data, together with the associated modified data structure, will be copied to one or more secondary storage systems and made available to application servers coupled to the secondary storage systems." Kodama at [0050]. "Turning now to the Figures, FIG. 1 is an illustration of a basic data processing system, designated generally with the reference numeral 10, employing the present invention. As FIG. 1 shows, the data processing system 10 may include a number of data centers, here illustrated by data centers 12A and 12N. The primary data center 12A includes a management console 14, a contents manager 16, and may or may not include an application server 18 and a storage system 22A. The other data centers 12, such as data center 12N, are application servers, each including a storage system 22N." Kodama at [0053].

Claim 7	Kodama
	 FIG_1 Kodama at FIG. 1. "The disk controllers 24 couple to storage media 34, 36, and preferably implements a "logical" storage in which one or more "volumes" map to magnetic disk storage units that implement the storage media 34, 36. Such logical storage makes it easier to respond to I/O requests from the servers (e.g., application servers 18, contents manager 16, etc.) of the data centers 12 using block level I/O to access the storage media. (As is conventional, an application process, executing for example on a server 18, will issue a file level I/O request to the file system of the server. In turn, the file system will issue the request as a block level I/O request to the storage system 24. While the storage system may not necessarily understand the data structure it can recognize the part that may be changed at a block level. This allows

Claim 7	Kodama
	the storage system 24 of the data center 12A to send those changes to the other data centers 12 according to the present invention.) A master volume 34 receives and maintains a master copy of all data of the system. Using the RC process 28, the disk controller 24 of the data center 12A will copy that data, and any subsequent changes to that data, to the replicated volumes 36 of the other data centers 12. Thus, the replicated volumes 36 mirror the data of the master volume 34." Kodama at [0055]. "The logical storage implemented by the disk controllers 24 of the storage systems 22 to receive block level I/O type requests that initiate accesses to the storage media 34, 36. The contents manager 16 maintains the necessary data structures to locate files and, using file level I/O requests to the file system, which converts them to block level I/O for access to those files on the storage media. The management console 14 provides a means to configure the system 10, including the placement of those files on the storage media of the storage systems 22." Kodama at [0056]. "Note that, as FIG. 1 shows, the application servers 18 are communicatively coupled to the contents manager by either a local area network (LAN2) connection, or a wide area network (WAN1) connection, such as the Internet. The storage systems 22 are separately connected to one another, using a wide area network WAN2, which also may be an Internet or other network connection. When changes are made to the data image maintained on the master volume 34, those changes are written to an unused area of storage. Then, the various file structures used by the file system to locate that data are modified, and the modified file structures also written to storage. The resulting data image and file structure changes are subsequently copied from the storage media 34 to the replicated volumes 36 at a block I/O level from the storage system 22A to the other storage systems 22B, using the WAN2 connection. Also, the necessary paths to that changed data will be sent in file level I/O format, using the LAN2 and WAN1 communication paths from the contents manager 16 to the application servers 18. (Bob: What do you want to mean by the necessary paths and file level I/O format? I can't understand

Claim 7	Kodama
	what this step means. The contents manager sends a message to the application servers via LAN and WAN1 and the message is used for flashing (in other word, invalidating) cached data in memory of the application servers so that the application servers don't use old data instead of new data in the secondary storage system.)" Kodama at [0057]. Pair Table "The pair table 30 is managed by the disk controllers 24, and includes such information a which volume is mirrored by which other volumes. and which volumes are associated with which storage systems. An example of the structure and content of a pair table is illustrated by FIG. 4, which shows a pair table 30′ as including in columns 56, 59, and 60, respectively containing a pair name, an identification of a primary volume associated with the pair name, and one or more secondary volumes. The pair name basically corresponds to the configuration name of the global mount points table. The primary volume is a volume which contains the original files, and is managed by the master file system process of the contents manager 16. The secondary volumes mirror the primary volume, and carry replications of the data of the primary volume. The secondary volumes are used by the application servers." Kodama at [0069]-[0070]. "As is conventional in file level systems, an inode will exist for each file in a file system, containing information for locating the file on the corresponding storage media 34 or 36 as well as properties related to the file. FIG. 6 is an illustration of representative inode information, showing that the file having an inode number 10, the file properties include a data when the file was last modified, the file size, a "New File" flag, and a list of directory ("Direct") entries. The New File flag is used by the master file system and shows whether the file exists when the file was opened. The directory entries (e.g., Direct 1, Direct 2, etc.) identify the locations of the various volumes data that form the file on the in the storage system. Each directory entry has associated with it a "New" flag which shows whether the

Claim 7	Kodama
	locations are used by a current inode of the file. The details about this flag will explained more fully below. Thus, for example, the file with the inode number of 10 was last updated on Dec. 31, 2005, is 100 megabytes in size, and its component parts can be found in the block 1, 2, . . . n in volume A1 on storage system A." Kodama at [0074].  *Inode Information* FIG_6 Kodama at FIG. 6. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks not previously contained in a root image and/or changes made by compute nodes to the

Claim 7	Kodama
	blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

Claim 7	Kodama

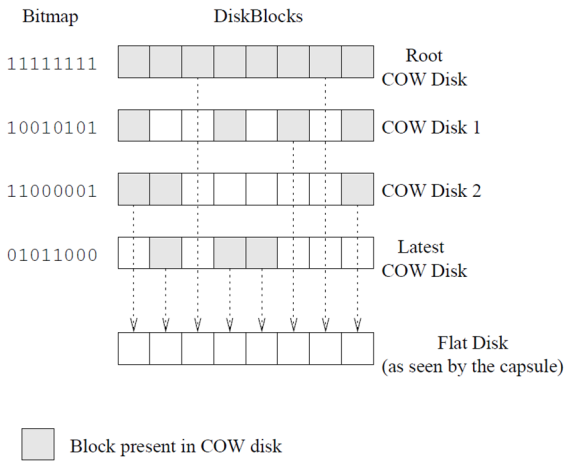


Figure 2: An example capsule disk.

Sapuntzakis at 6.

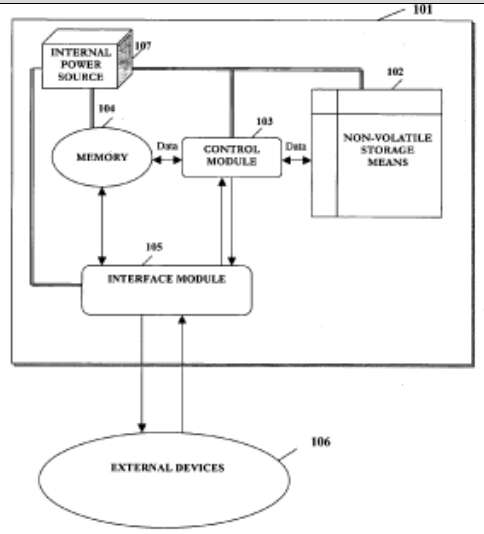
See also Sapuntzakis at 1-12.

"The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write

Claim 7	Kodama
	requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037].

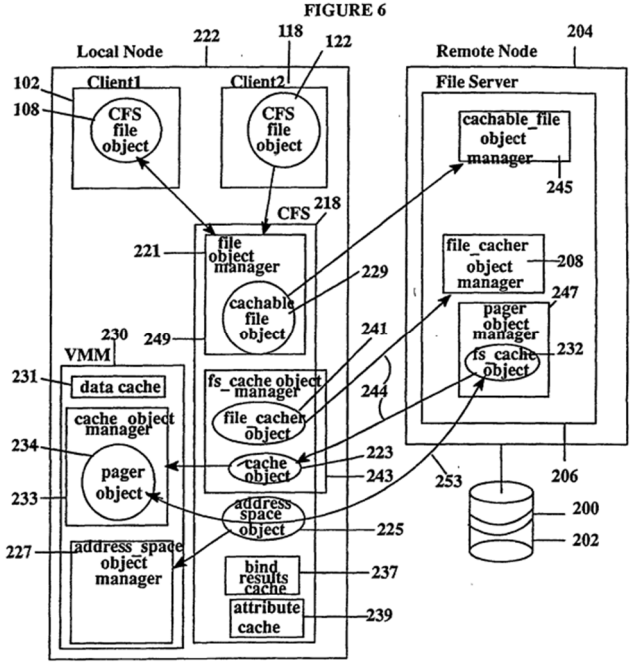
Claim 7	Kodama
	"The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"Returning to FIG. 16, if the requested file is not in the updated inode list, step 143 b will allocate a new inode number. Then, in step 144 b, the open procedure will create a new inode with the new inode number. Next, in step 145 b, the procedure will get the current inode of the file from a volume in a storage system or a cache memory. Then, in step 146 b, the procedure will copy the content of the current inode information to the new inode information. This will be followed by step 147 b where the new file flag is set in the new inode to "No." In step 148 b, a flag of each directory (direct) entry is set in the inode to "No." This entry informs one that the block specified by the directory entry is used in the current inode of the file. The procedure will then proceed to steps 150, 152." Kodama at [0109]. "This procedure is used by the configuration manager to update files in the updated inode list and is illustrated by FIG. 23. First, an update will change the directory list so that the application servers can use the new inodes of the updated files. After that all client file systems are told to invalidate their cached data corresponding to the directory list and the inode of the updated files. Without this step, there is the possibility that the client file system will continue to use outdated directory list and inode of an updated file. Finally, the master file system de-allocates and deletes information related to old inodes of the updated files." Kodama at [0133].

Claim 7	Kodama
	"Referring then to FIG. 23, in step 221, for new files in the updated inode list there will be added an entry in the corresponding directory list. For updated files in the updated inode list, the inode number of an entry in the corresponding directory list will be changed to the new inode number. In step 222 the blocks which include the modified directory lists will be flashed from the cache memory to a storage system, and in step 223 a wait is entered to allow the block to reach all of the storage systems. then, in step 224 an invalidation message is sent with the inode numbers of the updated files to each client file system. In step 225 the procedure waits for completion messages from all of the client file system." Kodama at [0134]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

Claim 7	Kodama
	 Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

Claim 7	Kodama
	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object

Claim 7	Kodama
	manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Kodama
	FIGURE 6  Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Kodama
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Kodama
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"Pair Table The pair table 30 is managed by the disk controllers 24, and includes such information a which volume is mirrored by which other volumes. and which volumes are associated with which storage systems. An example of the structure and content of a pair table is illustrated by FIG. 4, which shows a pair table 30′ as including in columns 56, 59, and 60, respectively containing a pair name, an identification of a primary volume associated with the pair name, and one or more secondary volumes. The pair name basically corresponds to the configuration name

Claim 11	Kodama
	of the global mount points table. The primary volume is a volume which contains the original files, and is managed by the master file system process of the contents manager 16. The secondary volumes mirror the primary volume, and carry replications of the data of the primary volume. The secondary volumes are used by the application servers." Kodama at [0069]-[0070]. "As is conventional in file level systems, an inode will exist for each file in a file system, containing information for locating the file on the corresponding storage media 34 or 36 as well as properties related to the file. FIG. 6 is an illustration of representative inode information, showing that the file having an inode number 10, the file properties include a data when the file was last modified, the file size, a "New File" flag, and a list of directory ("Direct") entries. The New File flag is used by the master file system and shows whether the file exists when the file was opened. The directory entries (e.g., Direct 1, Direct 2, etc.) identify the locations of the various volumes data that form the file on the in the storage system. Each directory entry has associated with it a "New" flag which shows whether the locations are used by a current inode of the file. The details about this flag will explained more fully below. Thus, for example, the file with the inode number of 10 was last updated on Dec. 31, 2005, is 100 megabytes in size, and its component parts can be found in the block 1, 2, . . . n in volume A1 on storage system A." Kodama at [0074]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or

Claim 11	Kodama
	Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop

Claim 11	Kodama
	preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows

Claim 11	Kodama
	the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-10

Invalidity of U.S. Patent No. 8,332,844 in View of "Implementing Scalable Disk-less Clusters using the Network File System" (NFS) ("Laros")

Laros titled "Implementing Scalable Disk-less Clusters using the Network File System" by James H. Laros et al. was published at least by January 2003.[1] Laros is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Laros does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Laros or combine the teachings of Laros with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

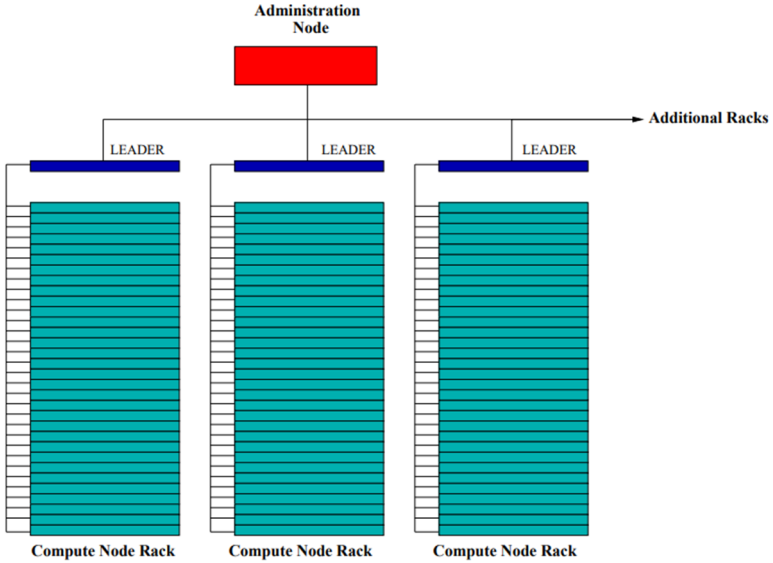
As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

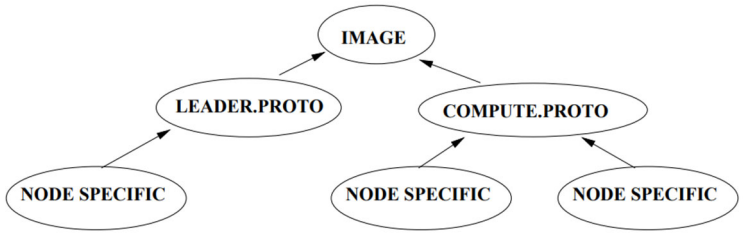
Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Laros
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Laros discloses "[t]his paper describes a methodology for implementing disk-less1 cluster systems using the Network File System (NFS) that scales to thousands of nodes. This method has been successfully deployed and is currently in use on several production systems at Sandia National Labs. This paper will outline our methodology and implementation, discuss hardware and software considerations in detail and present cluster configurations with performance numbers for various management operations like booting." Laros at Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"The hardware configuration used in our approach is hierarchical in nature. For the purposes of this paper the management network of the cluster will be described. The term management network describes the network, in this case Ethernet, used to support cluster infrastructure operations like booting the nodes of the cluster. At the top of the hierarchy is the administration node. This node contains the sole disk in the cluster system used for purposes of node initialization, export of the nfs-root filesystem, and subsequent cluster management. As shown in Figure 1, below the administration node are some number of secondary nodes we refer to as "leaders"." Laros at 4.

Claim 7	Laros
	 Laros at 3. "In practice, the administration node is itself a leader to the leader nodes. Think of a leader as a node that has some responsibility for other nodes, mainly in a support sense. Below the leader nodes are the nodes that make up the bulk of the cluster. These nodes satisfy various roles (the purpose the node serves in the overall system) predominantly serving as compute nodes (nodes where applications are executed). The ratio of leaders to administration node, and nodes to leaders is dependent on the type of system used to support these roles. For the purpose of this paper we will use a ratio of 32 nodes per leader. This ratio is used on both of the example configurations detailed in this paper. The nodes on each level of the hierarchy share a network with the node above it. For example, the administrative node shares a common network with all of the leader nodes. A leader node and the nodes under it also share a common network. While our methodology does not require a specific hardware configuration, this hardware hierarchy nicely accommodates the distribution of work necessary to facilitate our methodology on large scale systems." Laros at 4.

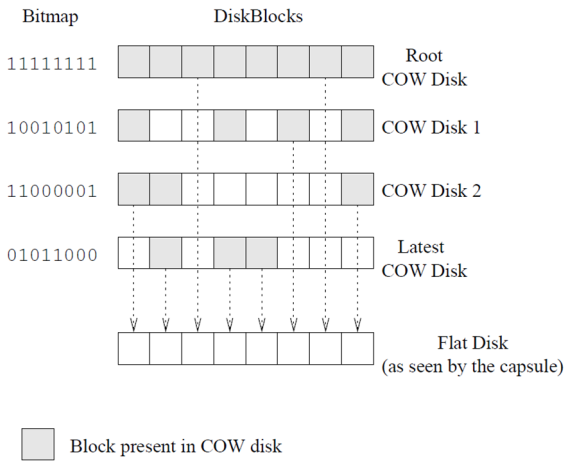
Claim 7	Laros
	"One of the pieces of information obtained from the leader by the compute nodes in the bootp exchange is typically the filename of the kernel to load. When the compute nodes request this file (kernel) the leader will service these requests through tftp daemons spawned in response to each request. When the first request for a kernel file is made the leader will not have this file cached and require the administration node to serve this file via NFS (along with the tftp daemon executable). Assuming that the compute nodes all require the same kernel file, which is often the case, this first request will be the only request that imposes on the administration node. All other requests by the initializing compute nodes will be serviced through the NFS cache resident in the leaders memory. These requests have to be made in a timely manner before the cache becomes stale, but normally we are trying to initialize the nodes as fast as possible." Laros at 6. "As mentioned, some files required by the disk-less cluster may be the same for every node in the cluster, which provides an opportunity for file sharing. Other files may be unique on a per-node basis. Our methodology allows a bootable hierarchy to be created that can exploit sharing at any level of commonality, and preserve a node level granularity when necessary, on a cluster-by-cluster basis." Laros at 6. "The first step in establishing the bootable hierarchy is to install what we call the image. The image is the stock Linux distribution installed in an alternate root path on the administration node. Rather than waste time selecting what files or packages will be installed and resolving conflicts based on this selection, we have taken the approach of installing all available software. This approach has no impact on the performance of the cluster (only files that are used are cached, see Section 4.2). While the image will be the same size regardless of the number of nodes in the cluster, the inode count increases as 6 the node count increases. The process of creating the bootable hierarchy, described in more detail below, heavily leverages linking. Since each link requires an inode, large clusters can impose a heavy inode requirement. This is an important consideration when preparing the filesystem that will hold the image and bootable hierarchy."

Claim 7	Laros
	Laros at 6. "The image is the master copy of the root file-system for the cluster. As the hierarchy is built on top of the image, files that are desired to be specific to a group of nodes, or a particular node will override the image. This allows for the wide range of file sharing or granularity down to the node level mentioned. For example, commands that exist in directories like /bin are most likely common to all nodes in the cluster regardless of the role they serve in the cluster. Configuration files such as files in /etc, that may determine specific characteristics like the IP address of a specific node's interface, are unique to each node. This leaves a wide area available for medial levels of grouping. Our implementation provides for intermediate grouping based on the role of the node. Other groupings based on any scheme desired are easily implemented. On our production systems we have implemented three role abstractions, but for the purposes of this paper we will only describe leader and compute roles. These groupings or abstractions are instantiated in prototype directories described in the following paragraphs. " Laros at 7. "In our test systems the nodes will serve the role of leader or compute. As mentioned, the leader nodes are nodes that provide infrastructure services to other nodes. The compute nodes make up the bulk of the cluster. It should be noted that the reason these two roles were selected is the likely file based commonality of nodes that serve these roles. If there is utility in grouping nodes by other roles, those roles could be used as the basis for grouping in addition to, or instead of compute and leader. Since it is likely that leader nodes, as a group, will share some common file requirements that may be different from the image, we construct a leader.proto directory to house these common differences. For the same purpose, we construct a compute.proto directory. Files that are specific to an individual node regardless of its role in the system will be placed in the node specific directory at the bottom of the hierarchy. Figure 2 illustrates a logical layout of the bootable hierarchy and the three levels of abstraction we will discuss." Laros at 7.

Claim 7	Laros
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"In our test systems the nodes will serve the role of leader or compute. As mentioned, the leader nodes are nodes that provide infrastructure services to other nodes. The compute nodes make up the bulk of the cluster. It should be noted that the reason these two roles were selected is the likely file based commonality of nodes that serve these roles. If there is utility in grouping nodes by other roles, those roles could be used as the basis for grouping in addition to, or instead of compute and leader. Since it is likely that leader nodes, as a group, will share some common file requirements that may be different from the image, we construct a leader.proto directory to house these common differences. For the same purpose, we construct a compute.proto directory. Files that are specific to an individual node regardless of its role in the system will be placed in the node specific directory at the bottom of the hierarchy. Figure 2 illustrates a logical layout of the bootable hierarchy and the three levels of abstraction we will discuss." Laros at 7.  Figure 2: Bootable Hierarchy logical layout Laros at 8. "The next level of abstraction in the bootable hierarchy and the final in our implementation is the node specific level. This level is provided to harbor node specific files. An area exists for each node in the cluster regardless of its role. This area may or may not contain anything specific to the node depending on the requirements imposed on that node in the cluster. In our implementation we find few files are required to be specific to an individual node, but this concept allows for a node to have a completely individualized root file-system if necessary. Figure 5 illustrates a file that is commonly unique to a specific node. The ifcfg-eth0 file in our example configuration contains

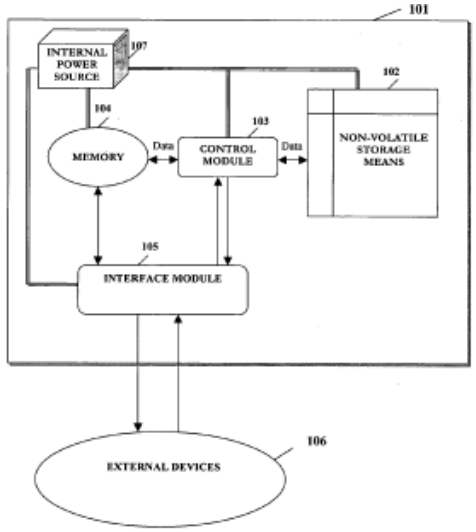
Claim 7	Laros
	information to initialize the 0th interface for the node. This file has node specific information like the IP address for the node. Since this information is specific to a node, in this case node.n-1.t-0, it is placed in the node specific directory for that node. Notice that this file does not exist in the image or the compute.proto path. This is because when the hierarchy was created it was anticipated that this file would be specific to the node level in the hierarchy. Even if this file did exist in either the image or proto levels in the hierarchy the effective file would still be the node specific file, due to the precedence rules of the bootable hierarchy." Laros at 9. **alternate−root−path/image/etc/sysocnfig/network−scripts** **alternate−root/path/compute.proto/etc/sysconfig/network−scripts/** **alternate−root−path/t−0/node.n−1.t−0/etc/sysconfig/network−scripts/ifcfg−eth0** Figure 5: ifcfg-eth0 file specific to node "To avoid unnecessary linking which also reduces the heavy inode requirement, we utilize this hierarchical linking only when it is beneficial. Entire directories like /bin, /lib and /usr, which so far have not required us to over-ride in any way, are mounted read-only and are shared by all the nodes in the cluster." Laros at 9. "Other directories that are a challenge to share like /dev and /var are simply located in the individual node specific directories. Since avoiding writes is critical to the success of this approach, a sparse /var skeleton is all that is required. Individual /dev directories can either contain a full complement of device entries, or can be paired down to reflect the individual requirements of a node. Section 7 discusses options like the device file-system which can be easily incorporated into this methodology and would remove the necessity of individual /dev directories." Laros at 10.

Claim 7	Laros
	To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

Claim 7	Laros
	 Figure 2: An example capsule disk. Sapuntzakis at 6. *See also* Sapuntzakis at 1-12. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write

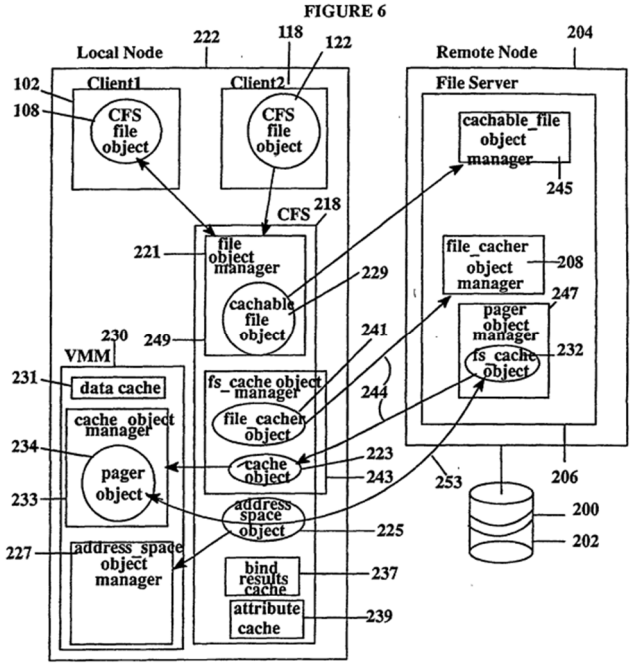
Claim 7	Laros
	requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037].

Claim 7	Laros
	"The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"One of the pieces of information obtained from the leader by the compute nodes in the bootp exchange is typically the filename of the kernel to load. When the compute nodes request this file (kernel) the leader will service these requests through tftp daemons spawned in response to each request. When the first request for a kernel file is made the leader will not have this file cached and require the administration node to serve this file via NFS (along with the tftp daemon executable). Assuming that the compute nodes all require the same kernel file, which is often the case, this first request will be the only request that imposes on the administration node. All other requests by the initializing compute nodes will be serviced through the NFS cache resident in the leaders memory. These requests have to be made in a timely manner before the cache becomes stale, but normally we are trying to initialize the nodes as fast as possible." Laros at 6. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with

Claim 7	Laros
	Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].  Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the

Claim 7	Laros
	component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module." Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind

Claim 7	Laros
	operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Laros
	 Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Laros
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Laros
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"The image is the master copy of the root file-system for the cluster. As the hierarchy is built on top of the image, files that are desired to be specific to a group of nodes, or a particular node will override the image. This allows for the wide range of file sharing or granularity down to the node level mentioned. For example, commands that exist in directories like /bin are most likely common to all nodes in the cluster regardless of the role they serve in the cluster. Configuration files such as files in /etc, that may determine specific characteristics like the IP address of a specific node's interface, are unique to each node. This leaves a wide area available for medial levels of grouping.

Claim 11	Laros
	Our implementation provides for intermediate grouping based on the role of the node. Other groupings based on any scheme desired are easily implemented. On our production systems we have implemented three role abstractions, but for the purposes of this paper we will only describe leader and compute roles. These groupings or abstractions are instantiated in prototype directories described in the following paragraphs. " Laros at 7. "In our test systems the nodes will serve the role of leader or compute. As mentioned, the leader nodes are nodes that provide infrastructure services to other nodes. The compute nodes make up the bulk of the cluster. It should be noted that the reason these two roles were selected is the likely file based commonality of nodes that serve these roles. If there is utility in grouping nodes by other roles, those roles could be used as the basis for grouping in addition to, or instead of compute and leader. Since it is likely that leader nodes, as a group, will share some common file requirements that may be different from the image, we construct a leader.proto directory to house these common differences. For the same purpose, we construct a compute.proto directory. Files that are specific to an individual node regardless of its role in the system will be placed in the node specific directory at the bottom of the hierarchy. Figure 2 illustrates a logical layout of the bootable hierarchy and the three levels of abstraction we will discuss." Laros at 7. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have

Claim 11	Laros
	required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session

Claim 11	Laros
	and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory."

Claim 11	Laros
	Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-11

Invalidity of U.S. Patent No. 8,332,844 in View of Thresher: An Efficient Storage Manager for Copy-on-write Snapshots ("Shrira")

Shrira titled "Thresher: An Efficient Storage Manager for Copy-on-write Snapshots" by Liuba Shrira et al. was published at least by May 30, 2006.[1] Shrira is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(a) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Shrira does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Shrira or combine the teachings of Shrira with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

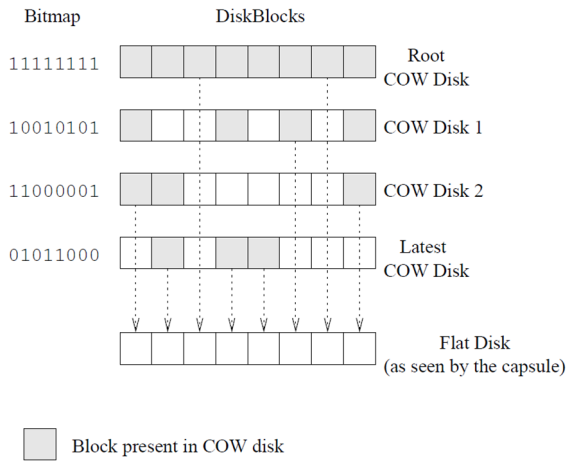
Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

Claim 7	Shrira
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Shrira discloses: "A new generation of storage systems exploit decreasing storage costs to allow applications to take snapshots of past states and retain them for long durations. Over time, current snapshot techniques can produce large volumes of snapshots. Indiscriminately keeping all snapshots accessible is impractical, even if raw disk storage is cheap, because administering such large-volume storage is expensive over a long duration. Moreover, not all snapshots are equally valuable. Thresher is a new snapshot storage management system, based on novel copy-on-write snapshot techniques, that is the first to provide applications the ability to discriminate among snapshots efficiently. Valuable snapshots can remain accessible or stored with faster access while less valuable snapshots are discarded or moved off-line. Measurements of the Thresher prototype indicate that the new techniques are efficient and scalable, imposing minimal (4%) performance penalty on expected common workloads." Shrira at Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"3 The snapshot system Thresher is implemented in SNAP [12], the snapshot system that provides snapshots for the Thor [7] object storage system. This section reviews the baseline storage and snapshot systems, using Figure 3 to trace their execution within Thresher. Our general approach to snapshot discrimination is applicable to snapshot systems that separate snapshots from the current storage system state. Such so-called split snapshot systems [16] rely on update-in-place storage and create snapshots by copying out the past states, unlike snapshot systems that rely on no-overwrite storage and do not separate snapshot and current states [13]. Split snapshots are attractive in long-lived systems because they allow creation of high-frequency snapshots without disrupting access to the current state while preserving the on-disk object clustering for the current state [12]. Our approach takes advantage of the separation between snapshot and current states to provide efficient

Claim 7	Shrira
	snapshot discrimination. We create a specialized snapshot representation tailored to the discrimination policy while copying out the past states." Shrira at 2.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"3.1 The storage system Thor has a client/server architecture. Servers provide persistent storage (called database storage) for objects. Clients cache copies of the objects and run applications that interact with the system by making calls to methods of cached objects. Method calls occur within the context of transaction. A transaction commit causes all modifications to become persistent, while an abort leaves no transaction changes in the persistent state. The system uses optimistic concurrency control [1]. A client sends its read and write object sets with modified object states to the server when the application asks to commit the transaction. If no conflicts were detected, the server commits the transaction." Shrira at 2. "An object belongs to a particular server. The object within a server is uniquely identified by an object reference (Oref). Objects are clustered into 8KB pages. Typically objects are small and there are many of them in a page. An object Oref is composed of a PageID and a oid. The PageID identifies the containing page and allows the lookup of an object location using a page table. The oid is an index into an offset table stored in the page. The offset table contains the object offsets within the page. This indirection allows us to move an object within a page without changing the references to it. When an object is needed by a transaction, the client fetches the containing page from the server. Only modified objects are shipped back to the server when the transaction commits. Thor provides transaction durability using the ARIES no-force no-steal redo log protocol [5]. Since only modified objects are shipped back at commit time, the server may need to do an installation read (iread) [8] to obtain the containing page from disk. An in-memory, recoverable cache called the modified object buffer(MOB) stores the committed modifications allowing to defer ireads and increase write absorption [4, 8]. The modifications are propagated to the

Claim 7	Shrira
	disk by a background cleaner thread that cleans the MOB. The cleaner processes the MOB in transaction log order to facilitate the truncation of the transaction log. For each modified object encountered, it reads the page containing the object from disk (iread) if the page is not cached, installs all modifications in the MOB for objects in that page, writes the updated page back to disk, and removes the objects from the MOB. The server also manages an in-memory page cache used to serve client fetch requests. Before returning a requested page to the client, the server updates the cache copy, installing all modifications in the MOB for that page so that the fetched page reflects the up-to-date committed state. The page cache uses LRU replacement but discards old dirty pages (it depends on ireads to read them back during MOB cleaning) rather than writing them back to disk immediately. Therefore the cleaner thread is the only component of the system that writes pages to disk." Shrira at 3. "3.2 Snapshots SNAP creates snapshots by copying out the past storage system states onto a separate snapshot archive disk A snapshot provides the same abstraction as the storage system, consisting of snapshot pages and a snapshot page table. This allows unmodified application code running in the storage system to run as BITE over a snapshot. SNAP copies snapshot pages and snapshot page table mappings into the archive during cleaning. It uses an incremental copy-on-write technique specialized for split snapshots: a snapshot page is constructed and copied into the archive when a page on the database disk is about to be overwritten the first time after a snapshot is declared. Archiving a page creates a snapshot page table mapping for the archived page." Shrira at 3. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary

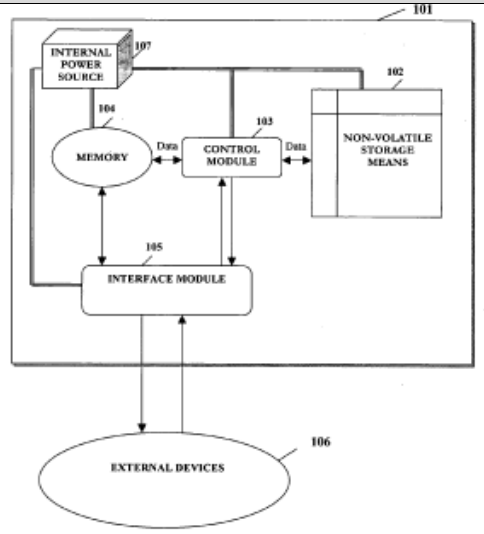
Claim 7	Shrira
	skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

Claim 7	Shrira
	 Figure 2: An example capsule disk. Sapuntzakis at 6. *See also* Sapuntzakis at 1-12. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write

Claim 7	Shrira
	requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037].

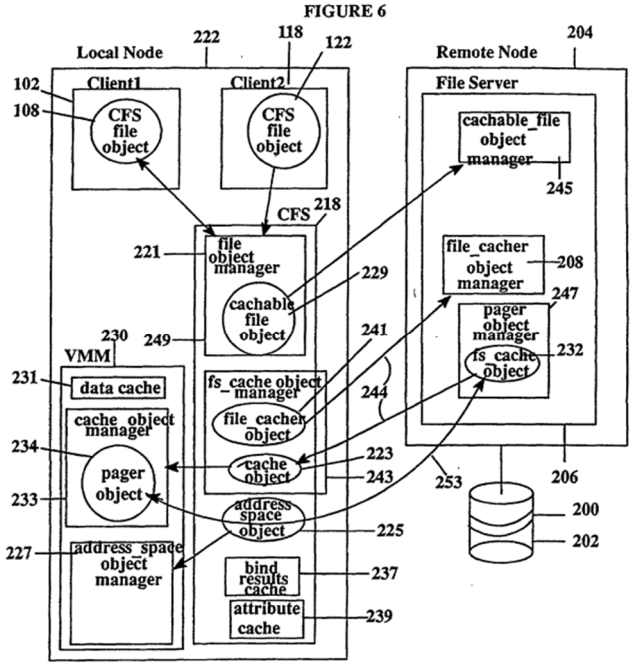
Claim 7	Shrira
	"The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"When an object is needed by a transaction, the client fetches the containing page from the server. Only modified objects are shipped back to the server when the transaction commits. Thor provides transaction durability using the ARIES no-force no-steal redo log protocol [5]. Since only modified objects are shipped back at commit time, the server may need to do an installation read (iread) [8] to obtain the containing page from disk. An in-memory, recoverable cache called the modified object buffer(MOB) stores the committed modifications allowing to defer ireads and increase write absorption [4, 8]. The modifications are propagated to the disk by a background cleaner thread that cleans the MOB. The cleaner processes the MOB in transaction log order to facilitate the truncation of the transaction log. For each modified object encountered, it reads the page containing the object from disk (iread) if the page is not cached, installs all modifications in the MOB for objects in that page, writes the updated page back to disk, and removes the objects from the MOB." Shrira at 3. "The server also manages an in-memory page cache used to serve client fetch requests. Before returning a requested page to the client, the server updates the cache copy, installing all modifications in the MOB for that page so that the fetched page reflects the up-to-date committed state. The page cache uses LRU replacement but discards old dirty pages (it depends on ireads to read them back during MOB

Claim 7	Shrira
	cleaning) rather than writing them back to disk immediately. Therefore the cleaner thread is the only component of the system that writes pages to disk." Shrira at 3. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

Claim 7	Shrira
	 Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

Claim 7	Shrira
	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object

Claim 7	Shrira
	manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Shrira
	 Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Shrira
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Shrira
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"When an object is needed by a transaction, the client fetches the containing page from the server. Only modified objects are shipped back to the server when the transaction commits. Thor provides transaction durability using the ARIES no-force no-steal redo log protocol [5]. Since only modified objects are shipped back at commit time, the server may need to do an installation read (iread) [8] to obtain the containing page from disk. An in-memory, recoverable cache called the modified object buffer(MOB) stores the committed modifications allowing to defer ireads and increase write absorption [4, 8]. The modifications are propagated to the

Claim 11	Shrira
	disk by a background cleaner thread that cleans the MOB. The cleaner processes the MOB in transaction log order to facilitate the truncation of the transaction log. For each modified object encountered, it reads the page containing the object from disk (iread) if the page is not cached, installs all modifications in the MOB for objects in that page, writes the updated page back to disk, and removes the objects from the MOB." Shrira at 3. "3.2 Snapshots SNAP creates snapshots by copying out the past storage system states onto a separate snapshot archive disk A snapshot provides the same abstraction as the storage system, consisting of snapshot pages and a snapshot page table. This allows unmodified application code running in the storage system to run as BITE over a snapshot. SNAP copies snapshot pages and snapshot page table mappings into the archive during cleaning. It uses an incremental copy-on-write technique specialized for split snapshots: a snapshot page is constructed and copied into the archive when a page on the database disk is about to be overwritten the first time after a snapshot is declared. Archiving a page creates a snapshot page table mapping for the archived page." Shrira at 3. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have

Claim 11	Shrira
	required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session

Claim 11	Shrira
	and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013].

Claim 11	Shrira
	"The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-12

Invalidity of U.S. Patent No. 8,332,844 in View of PCT Patent Application Publication No. WO 2006/074869 ("Bayer")

Bayer titled "Data storage system and method for operation thereof" to Rudolf Bayer was filed on January 4, 2006 and published on July 20, 2006.[1] Bayer is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent.

To the extent Plaintiff alleges Bayer does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Bayer or combine the teachings of Bayer with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

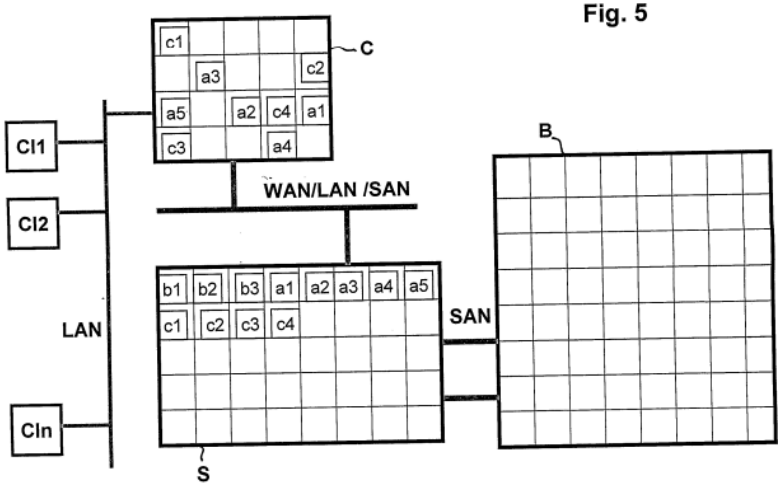
As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

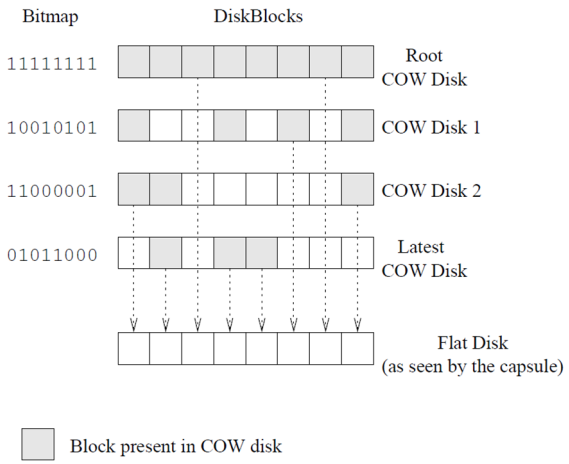
Claim 7	Bayer
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Bayer discloses: "The invention relates to a data storage system with a number of user computers (Clients) (C11, C12,..., C1n), a storage system for storing of files (a, b, c), access to which is achieved by the user computers (C11, C12, ...,C1n), a back-up system (B), for securing files stored in the storage system, whereby the storage system comprises a cache unit (C) and a storage unit (S). Said cache unit (C) comprises at least one cache, connected to one or more user computers (C11, C12,..., C1n) and to the storage unit (S) and embodied to only store those files required for processing by one or more user computers (C11, C12, ..., C1n). The storage unit (S) is connected to the cache unit (C) and to the backup system (B) and is embodied to permanently store all files." Bayer at Abstract.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	"Figure 5 shows a first embodiment of a possible architecture of a data storage system according to the invention with a plurality of user computers CIl, C12...CIn. The user computers CIl to CIn are connected to the cache unit C via a LAN (Local Area Network) (in the cache unit C, the two files a and c are stored page by page in blocks of the cache unit, similar to the representation in Figure 4). The cache unit C is in turn connected via another network connection (LAN, SAN or WAN or similar) to the storage unit S, in which the three files a, b, c are permanently stored sequentially as shown in Figure 4. Finally, the storage unit S is connected via a WAN, a LAN or a storage area network SAN to a backup system B for securing data stored in the storage unit S." Bayer at [0026].

Claim 7	Bayer
	Fig. 5  "The storage unit S is thus arranged between the cache unit C and the backup system B. The storage unit S does not require a block interface, but only a simple so-called file interface, i.e. only entire files (or very large parts of them, which are significantly larger than a single page of the file) are transferred between the storage unit and the cache unit or between the storage unit and the backup system." Bayer at [0028]. "New files are first created in the cache unit, and the created file is then permanently stored in the storage unit. It is particularly advantageous to store the files sequentially when transferring them from the cache unit to the storage unit. The file can then be deleted from the cache unit C immediately or at a later time that can be set or determined by rules (e.g. upon receipt of an invalidation message if the file has been modified in the meantime by another user (see below))." Bayer at [0029]. "If a user wishes to re-edit a file that has already been created and stored in the storage unit S, the corresponding file is transferred to the cache unit for editing by the user. After the file has been processed (updated) on the cache unit, the file is

Claim 7	Bayer
	transferred as a whole from the cache unit to the storage unit and stored in the storage unit (preferably sequentially). The file can be saved in the storage unit at the same location as before, especially if the size of the file has not changed. Most of the time, however, saving will take place at a different location." Bayer at [0030].
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	"According to the invention, the previous file server is divided into two independent and only very weakly coupled subsystems, namely the cache unit C and the storage unit S. The permanent storage of files takes place, as described, in the storage unit S and the files are also backed up from the storage unit S. In order to secure the contents of the storage unit, entire files or difference files, so-called . Deltas are transferred from the storage unit S to the backup system B via a file interface. If files are no longer required, they can be moved to the backup system B for the purpose of long-term archiving or to fulfill any legally prescribed retention obligations (compliance rules) and then deleted from the storage unit S. A directory of the archived files can be stored in the storage unit S, although this directory is preferably also stored in the backup system." Bayer at [0031]. As mentioned above, files can generally be deleted in the cache unit if the closed file has been moved from the cache unit to the storage unit. In order to optimize the so-called performance of the overall system, files can also remain in the cache unit, which saves a transfer back from the storage unit to the cache unit when a user requests that a file be edited again. However, this is not necessary for the invention to function correctly (therefore, according to the invention, the contents of the cache unit do not have to be saved in the backup system). Bayer at [0039]. "In addition, the functionality and performance of the storage unit S can be improved as follows: The storage unit S can store several versions of a file a that are created in succession, where the versions are referred to as a . l , a .2 , ... , a . n . The physical

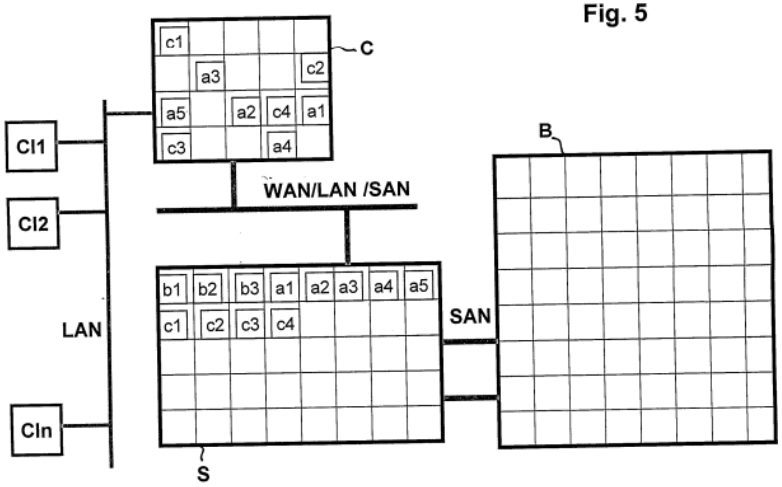
Claim 7	Bayer
	storage of these logical versions a . i-1 , a . i, a . i+1 ... is optimized by the xor difference formation known to the person skilled in the art in such a way that only the compressed differences a_delta . i-l , a_delta . i and the most recent file version (here: a . i+1) are physically stored. The differences arise according to the following relationships: a_delta . i = a. i xor a . i+1 for i = 1 , 2 , ... and are computed in the storage unit S when a new version a . i+1 is moved from C to S, since at this time the version a . i is still present in S . The earlier versions of the files a . i and a . i-1 can be calculated very quickly if required using the following relationship: a. i = a. i+1 xor a_delta . i a . i-1 = a. i xor a_delta . i-l For example, if a total of 4 versions of a are to be kept in the storage unit, the following files are required: a_delta . i-3 , a_delta . i-2 , a_delta . i-l , a . i . Bayer at [0050]-[0056]. "At the time a . i+1 is transferred to the storage system S, the file a . i still exists in the backup system B or it can be reconstructed based on saved delta versions. Therefore, it is sufficient to transfer only the file difference a_delta . i between a . i and a . i+1 from S to B in order to save the new file version a . i+1. In this way, the transfer volume between S and B or between C and S can be significantly reduced. This method is particularly advantageous when there is a radio connection between the network (as for example in the embodiments of Figure 13 or Figure 14). The storage unit S additionally contains a multidimensional meta-database and a corresponding database management system to manage all files and their delta differences and to be able to reconstruct files according to their state at any earlier point in time. This gives the storage unit additional capabilities of a so-called. Information Lifecycle Management Systems (ILMS)." Bayer at [0057]-[0058]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary

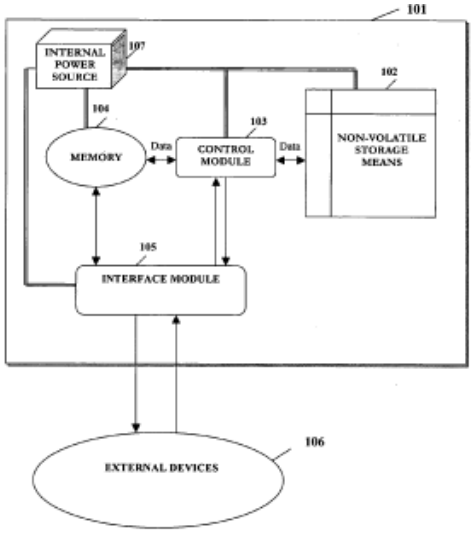
Claim 7	Bayer
	skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

Claim 7	Bayer
	 Figure 2: An example capsule disk. Sapuntzakis at 6. *See also* Sapuntzakis at 1-12. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write

Claim 7	Bayer
	requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037].

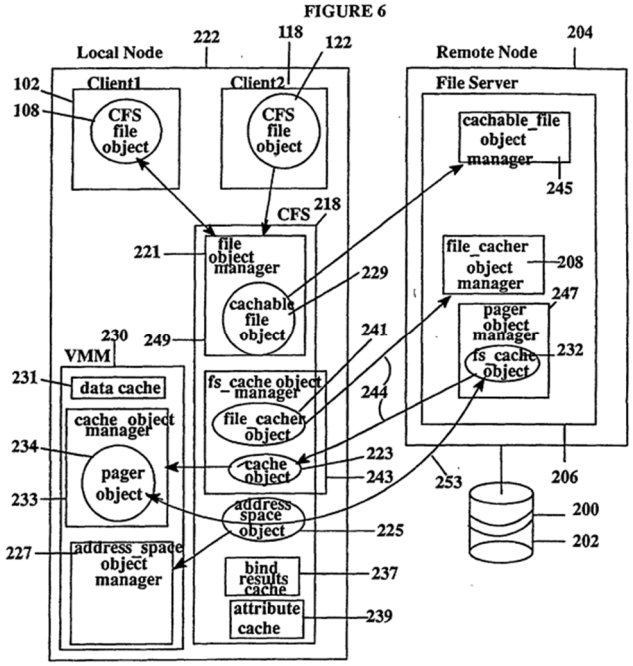
Claim 7	Bayer
	"The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	"Figure 5 shows a first embodiment of a possible architecture of a data storage system according to the invention with a plurality of user computers CIl, C12...CIn. The user computers CIl to CIn are connected to the cache unit C via a LAN (Local Area Network) (in the cache unit C, the two files a and c are stored page by page in blocks of the cache unit, similar to the representation in Figure 4). The cache unit C is in turn connected via another network connection (LAN, SAN or WAN or similar) to the storage unit S, in which the three files a, b, c are permanently stored sequentially as shown in Figure 4. Finally, the storage unit S is connected via a WAN, a LAN or a storage area network SAN to a backup system B for securing data stored in the storage unit S." Bayer at [0026].

Claim 7	Bayer
	Fig. 5  "The storage unit S is thus arranged between the cache unit C and the backup system B. The storage unit S does not require a block interface, but only a simple so-called file interface, i.e. only entire files (or very large parts of them, which are significantly larger than a single page of the file) are transferred between the storage unit and the cache unit or between the storage unit and the backup system." Bayer at [0028]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would

Claim 7	Bayer
	have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].  Fig. 1 Holzmann at FIG. 1 "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage

Claim 7	Bayer
	means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." Holzmann at [0017]. "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module." Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the

Claim 7	Bayer
	fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].

Claim 7	Bayer
	 Nelson at FIG. 6. *See also* Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4. "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for example, information about what

Claim 7	Bayer
	files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].

Claim 11	Bayer
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	"At the time a . i+1 is transferred to the storage system S, the file a . i still exists in the backup system B or it can be reconstructed based on saved delta versions. Therefore, it is sufficient to transfer only the file difference a_delta . i between a . i and a . i+1 from S to B in order to save the new file version a . i+1. In this way, the transfer volume between S and B or between C and S can be significantly reduced. This method is particularly advantageous when there is a radio connection between the network (as for example in the embodiments of Figure 13 or Figure 14).

Claim 11	Bayer
	The storage unit S additionally contains a multidimensional meta-database and a corresponding database management system to manage all files and their delta differences and to be able to reconstruct files according to their state at any earlier point in time. This gives the storage unit additional capabilities of a so-called. Information Lifecycle Management Systems (ILMS)." Bayer at [0057]-[0058]. To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6.

Claim 11	Bayer
	"Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified

Claim 11	Bayer
	by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory."

Claim 11	Bayer
	Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-13

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2004/0153639 ("Cherian")

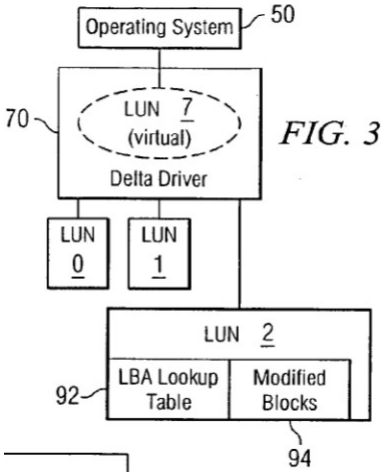
Cherian titled "System and method for sharing storage to boot multiple servers" to Jacob Cherian was filed on February 5, 2003 and published on August 5, 2004.[1] Cherian is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

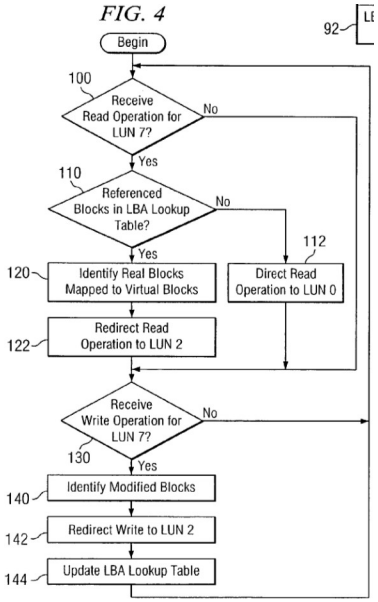
To the extent Plaintiff alleges Cherian does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Cherian or combine the teachings of Cherian with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	**Prior Art**
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Cherian discloses "[a] method for providing data to a plurality of compute nodes." For example: Cherian at Figure 3.  *FIG. 3* Cherian at Figure 4.

Claims	Prior Art
	 Cherian at [0001] The present invention relates in general to information handling systems and, in particular, to systems and methods for sharing storage to boot multiple information handling systems.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Cherian discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: Cherian at Figure 3. Cherian at Figure 4. Cherian at [0014] the servers may be connected to shared storage using any appropriate technology, including technologies supporting block storage protocols

Claims	Prior Art
	Cherian at [0021] In particular, in the example embodiment shown in FIG. 3, delta driver 70 provides a virtual storage subdivision (e.g., a virtual operating-system LUN) for booting the operating system into server 20, based on (a) a shared operating-system storage subdivision in shared storage 40 containing substantially unmodified operating-system data and (b) a delta storage subdivision in shared storage 40 containing modified operating-system data configured for server 20. Cherian at [0024] As shown in FIG. 3, the physical storage subdivisions in modular computing system 10 include a single, shared operating-system LUN 0 that contains the operating system code and software that is common to all servers 20, 22, 24. LUN 0 may be designated as read-only, Cherian at [0032] As described in greater detail below with reference to FIG. 4, delta drivers may provide respective virtual storage subdivisions for the different instances of the operating system executing in the different servers by automatically performing redirect-on-write and redirect-on-read operations. For instance, delta driver 70 may automatically access storage addresses in delta storage subdivision 2 in response to operations from server 20 involving the modified operating-system data configured for server 20. And, for operations from server 20 involving substantially unmodified operating-system data, delta driver 70 may automatically access storage addresses in shared operating-system storage subdivision 0.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	Cherian discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." For example: Cherian at Figure 3. Cherian at Figure 4. Cherian at [0021]

Claims	Prior Art
	In particular, in the example embodiment shown in FIG. 3, delta driver 70 provides a virtual storage subdivision (e.g., a virtual operating-system LUN) for booting the operating system into server 20, based on (a) a shared operating-system storage subdivision in shared storage 40 containing substantially unmodified operating-system data and (b) a delta storage subdivision in shared storage 40 containing modified operating-system data configured for server 20. Cherian at [0027] In the example embodiment, shared storage 40 also includes a distinct delta storage subdivision for each server in modular computing system 10. For instance, in the example embodiment, delta LUN 2 in FIG. 3 is the delta storage subdivision for server 20. Consequently, as described in greater detail below, delta LUN 2 may include modified blocks 94 of data from operating-system LUN 0, such as operating-system configuration data written to virtual LUN 7 by server 20 and redirected to delta LUN 2 by delta driver 70. Cherian at [0032] As described in greater detail below with reference to FIG. 4, delta drivers may provide respective virtual storage subdivisions for the different instances of the operating system executing in the different servers by automatically performing redirect-on-write and redirect-on-read operations. For instance, delta driver 70 may automatically access storage addresses in delta storage subdivision 2 in response to operations from server 20 involving the modified operating-system data configured for server 20. And, for operations from server 20 involving substantially unmodified operating-system data, delta driver 70 may automatically access storage addresses in shared operating-system storage subdivision 0. Cherian at [0041] As depicted at block 130, delta driver 70 may then determine whether it has received a write operation for virtual storage subdivision 7, for example after processing a read operation or determining at block 100 that a read operation has not been received. If no write operation has been received, the process may simply return to block 100 to await the next I/O command. If a write operation for storage subdivision 7 has been received, delta driver 70 may identify the blocks in delta storage subdivision 2 that will receive the data from the write command, as shown at block 140. Delta driver 70 may also redirect the command to delta storage subdivision 2, as depicted at block 142, and update LBA lookup table 92 to associate the modified blocks in delta storage subdivision 2 with the corresponding blocks in virtual storage subdivision 7. Delta driver 70 may thus automatically update the delta driver configuration data with a storage address in delta storage subdivision 2 for the redirected write operation. The process may then return to block 100, with subsequent I/O commands processed in a manner more or less like that described above.

Claims	Prior Art
	Cherian at claim 4 4. The information handling system of claim 1, wherein, in response to a write operation from the first server that involves the substantially unmodified operating-system data, the delta driver automatically performs operations comprising: redirecting the write operation to the first delta storage subdivision;
[11.0] 11. The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	Cherian discloses "[t]he method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments." For example: Cherian at [0032] As described in greater detail below with reference to FIG. 4, delta drivers may provide respective virtual storage subdivisions for the different instances of the operating system executing in the different servers by automatically performing redirect-on-write and redirect-on-read operations. For instance, delta driver 70 may automatically access storage addresses in delta storage subdivision 2 in response to operations from server 20 involving the modified operating-system data configured for server 20. And, for operations from server 20 involving substantially unmodified operating-system data, delta driver 70 may automatically access storage addresses in shared operating-system storage subdivision 0.

Exhibit 844-14

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2007/0057958 ("Bucher")

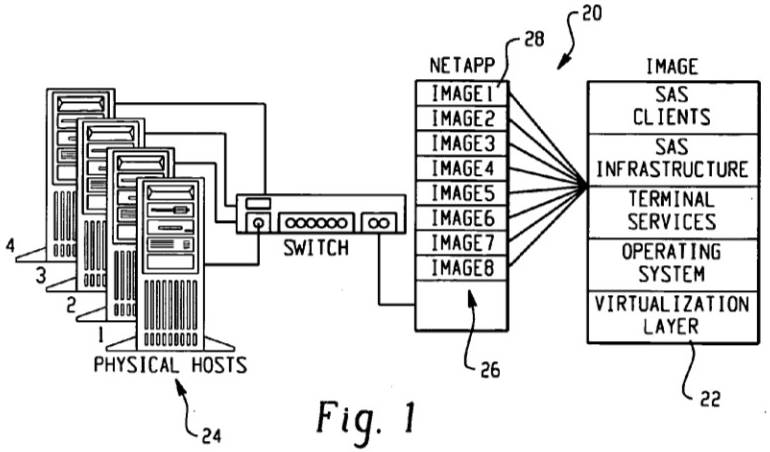
Bucher titled "Computer-implemented systems and methods for managing images" to Cristopher J. Bucher was filed on August 9, 2006 and published on March 15, 2007.[1] Bucher is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. § 102(e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

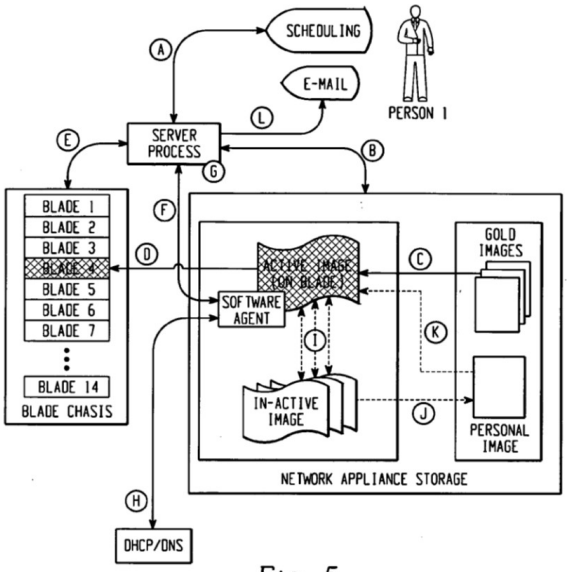
To the extent Plaintiff alleges Bucher does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Bucher or combine the teachings of Bucher with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

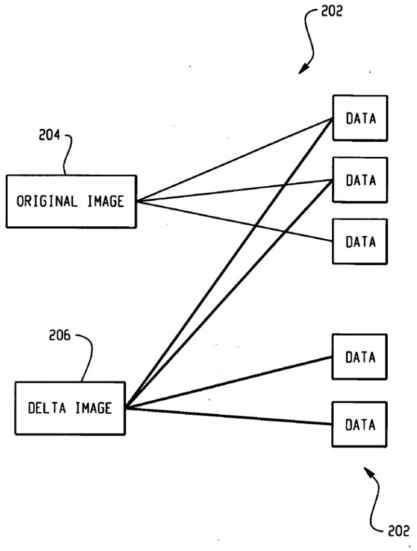
As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	**Prior Art**
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Bucher discloses "[a] method for providing data to a plurality of compute nodes." For example: Bucher at Figure 1.  Fig. 1 Bucher at Figure 5

Claims	Prior Art
	 Fig. 5 Bucher at [0022] FIG. 4 depicts at 100 two types of images involved in an image loading process: permanent/gold images 102 and working images 104 (e.g., sets of pointers). A permanent/gold image does not get mapped to a server (e.g., a blade server or blade chassis 106). Its purpose is to be the "original" copy of an image (e.g., SAS Demo application). As an illustration, it holds the thirty Gigabytes worth of information that makes up the "C:" drive. To make use of a permanent image, a working version 110 is created based upon the permanent image. Working images 104 do not consume space until an entity (e.g., a user or a software application) starts to make changes on the server. Only the changes are saved to the working image, thereby not consuming thirty Gigabytes per machine that is in use.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Bucher discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: Bucher at Figure 6

Claims	Prior Art
	 *Fig. 6* Bucher at [0022] FIG. 4 depicts at 100 two types of images involved in an image loading process: permanent/gold images 102 and working images 104 (e.g., sets of pointers). A permanent/gold image does not get mapped to a server (e.g., a blade server or blade chassis 106). Its purpose is to be the "original" copy of an image (e.g., SAS Demo application). As an illustration, it holds the thirty Gigabytes worth of information that makes up the "C:" drive. To make use of a permanent image, a working version 110 is created based upon the permanent image. Working images 104 do not consume space until an entity (e.g., a user or a software application) starts to make changes on the server. Only the changes are saved to the working image, thereby not consuming thirty Gigabytes per machine that is in use. Bucher at [0027] FIG. 6 depicts at 200 an approach for how images can be stored on a system, such as in a NetAPP system. The data blocks 202 represent pieces of an image written to disk (similar to sectors). Bucher at [0027] The original image 204 contains a list of pointers to the data blocks that make it up. The delta image 206 (e.g., a working image that has been cloned from the original) contains a list of pointers that may contain none, one or more, or all of the same pointers as the original plus a list of pointers to new data that was

Claims	Prior Art
	written as the image changed. New data can be pieces that replace pieces from the original or it can be new data added to the image. For example, a user updating a SAS dataset may cause one or more new data blocks to get written. The delta image 206 removes the pointers to the old data blocks that are no longer valid and creates new pointers to the newly written data. The original data blocks are not deleted because the original image still references them.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	Bucher discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." For example: Bucher at Figure 6 Bucher at [0022] FIG. 4 depicts at 100 two types of images involved in an image loading process: permanent/gold images 102 and working images 104 (e.g., sets of pointers). A permanent/gold image does not get mapped to a server (e.g., a blade server or blade chassis 106). Its purpose is to be the "original" copy of an image (e.g., SAS Demo application). As an illustration, it holds the thirty Gigabytes worth of information that makes up the "C:" drive. To make use of a permanent image, a working version 110 is created based upon the permanent image. Working images 104 do not consume space until an entity (e.g., a user or a software application) starts to make changes on the server. Only the changes are saved to the working image, thereby not consuming thirty Gigabytes per machine that is in use. Bucher at [0027] FIG. 6 depicts at 200 an approach for how images can be stored on a system, such as in a NetAPP system. The data blocks 202 represent pieces of an image written to disk (similar to sectors). Bucher at [0027] The original image 204 contains a list of pointers to the data blocks that make it up. The delta image 206 (e.g., a working image that has been cloned from the original) contains a list of pointers that may contain none, one or more, or all of the same pointers as the original plus a list of pointers to new data that was written as the image changed. New data can be pieces that replace pieces from the original or it can be

Claims	Prior Art
	new data added to the image. For example, a user updating a SAS dataset may cause one or more new data blocks to get written. The delta image 206 removes the pointers to the old data blocks that are no longer valid and creates new pointers to the newly written data. The original data blocks are not deleted because the original image still references them. Bucher at [0037] When a reservation is activated, the system makes a "copy-on-write" copy of the specified image. This means that a copy is created (that will only contain the changes) as a mountable entity. The system attaches this mountable copy to the specific blade. The system then powers on the blade, which boots as normal.

Exhibit 844-15

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 7,197,608 ("Mikuma")

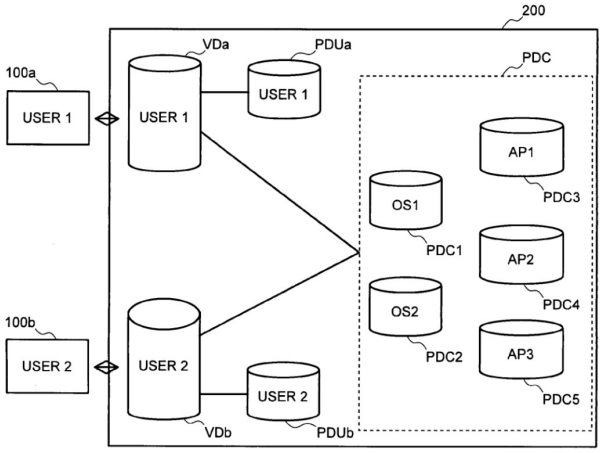
Mikuma titled "Software management method for a storage system, and storage system" to Ayumi Mikuma et al. was filed on July 12, 2004 and issued on March 27, 2007.[1] Mikuma is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

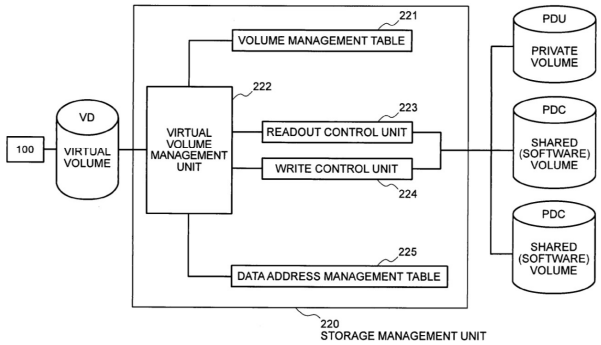
To the extent Plaintiff alleges Mikuma does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Mikuma or combine the teachings of Mikuma with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	Prior Art
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Mikuma discloses "[a] method for providing data to a plurality of compute nodes." For example: Mikuma at 1:15–17 The present invention relates to a software management method of a mass storage device that is accessed by a plurality of clients.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Mikuma discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: Mikuma at Figure 2. FIG.2  Mikuma at Figure 3.

Claims	Prior Art
	FIG.3  Mikuma at 1:67–2:10 The mass storage device comprises a shared volume that is shared by a plurality of users, and private volumes that contain data specific to each user. The mass storage device links these shared volumes and private volumes and allocates these volumes to each user as virtual volumes. The mass storage device manages, with respect to the shared volume and private volumes, the association between the users using the software contained in the shared volume and the private volumes used by the users, and stores setting information for each piece of software in the private volume. Mikuma at 4:35–42 In this embodiment, storage in the shared volumes PDC is assumed to be as follows: an operating system (hereinafter 'OS') 1 is stored in PDC1, an OS 2 in PDC2, an application program (hereinafter 'AP') 1 in PDC3, an AP2 in PDC4, and an AP3 in PDC5. Further, the VDa is associated with the private volume PDUa and shared volumes PDC. VDb is associated with the private volume PDUb and shared volumes PDC. Mikuma at 4:44–50 The client 100 transmits a request to the storage device 200 to designate a virtual volume VD and read data at a variety of times such as during remote booting. The storage device 200 selects and reads data corresponding to the virtual volume VD designated by the read command from the private volume PDU or shared volume PDC and sends this data back to the client 100. Mikuma at 5:35–50 In this embodiment, the readout control block 223 follows predetermined rules when reading data from the volumes PD corresponding with the virtual volume VD. Here, 'predetermined rules' are rules

Claims	Prior Art
	according to which the search for the private volume PDUa is prioritized over that for the shared volumes PDC1 and PDC3 in the volume PD corresponding with the virtual volume VD when the client 100 retrieves (reads) data in the virtual volume VD, for example. As a result, the storage device 200 is able to prioritize the search for client-specific setting information of an application program, OS, or the like (that is, setting information for a default not used by anyone is not shown to a client that already has setting information). Further, the readout control block 223 communicates the value of the address of the private volume PDU afforded search priority, to the virtual volume management block 222.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	Mikuma discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." For example: Mikuma at Figure 2. Mikuma at Figure 3. Mikuma at 1:67–2:10 The mass storage device comprises a shared volume that is shared by a plurality of users, and private volumes that contain data specific to each user. The mass storage device links these shared volumes and private volumes and allocates these volumes to each user as virtual volumes. The mass storage device manages, with respect to the shared volume and private volumes, the association between the users using the software contained in the shared volume and the private volumes used by the users, and stores setting information for each piece of software in the private volume. Mikuma at 2:24–35 the mass storage device sets the virtual volumes so that the data writing of software setting information generated for each shared volume by the computer used by the user is transferred to the private volume of the corresponding user. Further, the mass storage device provides each volume with a search order for searching for data for which a read designation is received by the user, thereby rendering a hierarchical structure. More

Claims	Prior Art
	specifically, the mass storage device prioritizes the search for the private volume corresponding with the user that issued the read command over the search for the shared volume where the software is stored. Mikuma at 4:35–42 In this embodiment, storage in the shared volumes PDC is assumed to be as follows: an operating system (hereinafter 'OS') 1 is stored in PDC1, an OS 2 in PDC2, an application program (hereinafter 'AP') 1 in PDC3, an AP2 in PDC4, and an AP3 in PDC5. Further, the VDa is associated with the private volume PDUa and shared volumes PDC. VDb is associated with the private volume PDUb and shared volumes PDC. Mikuma at 4:44–50 The client 100 transmits a request to the storage device 200 to designate a virtual volume VD and read data at a variety of times such as during remote booting. The storage device 200 selects and reads data corresponding to the virtual volume VD designated by the read command from the private volume PDU or shared volume PDC and sends this data back to the client 100. Mikuma at 5:35–50 In this embodiment, the readout control block 223 follows predetermined rules when reading data from the volumes PD corresponding with the virtual volume VD. Here, 'predetermined rules' are rules according to which the search for the private volume PDUa is prioritized over that for the shared volumes PDC1 and PDC3 in the volume PD corresponding with the virtual volume VD when the client 100 retrieves (reads) data in the virtual volume VD, for example. As a result, the storage device 200 is able to prioritize the search for client-specific setting information of an application program, OS, or the like (that is, setting information for a default not used by anyone is not shown to a client that already has setting information). Further, the readout control block 223 communicates the value of the address of the private volume PDU afforded search priority, to the virtual volume management block 222. Mikuma at 5:51–56 Further, the write control block 224 also follows the rule above. That is, this is a rule according to which a request from the client 100a to write to a data region that corresponds to shared volumes PDC1 and PDC3 allocated to the virtual volume VDa is transferred to and executed by the private volume PDUa. Mikuma at 14:60–64 Although the computer 100 was described in the embodiment above by using a diskless computer, a computer with a built-in disk device may actually be used. In this case, the internal disk of the computer can be used as the private volume PDU, cache region, and so forth.

Claims	Prior Art
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Mikuma discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: Mikuma at 14:60–64 Although the computer 100 was described in the embodiment above by using a diskless computer, a computer with a built-in disk device may actually be used. In this case, the internal disk of the computer can be used as the private volume PDU, cache region, and so forth.
[11.0] 11. The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	Mikuma discloses "[t]he method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments." For example: Mikuma at Figure 3. Mikuma at 4:44–50 The client 100 transmits a request to the storage device 200 to designate a virtual volume VD and read data at a variety of times such as during remote booting. The storage device 200 selects and reads data corresponding to the virtual volume VD designated by the read command from the private volume PDU or shared volume PDC and sends this data back to the client 100. Mikuma at 5:35–50 In this embodiment, the readout control block 223 follows predetermined rules when reading data from the volumes PD corresponding with the virtual volume VD. Here, 'predetermined rules' are rules according to which the search for the private volume PDUa is prioritized over that for the shared volumes PDC1 and PDC3 in the volume PD corresponding with the virtual volume VD when the client 100 retrieves (reads) data in the virtual volume VD, for example. As a result, the storage device 200 is able to prioritize the search for client-specific setting information of an application program, OS, or the like (that is, setting information for a default not used by anyone is not shown to a client that already has setting information). Further, the readout control block 223 communicates the value of the address of the private volume PDU afforded search priority, to the virtual volume management block 222.

Exhibit 844-16

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2005/0283575 ("Kobayashi")

Kobayashi titled "Information storing method for computer system including a plurality of computers and storage system" to Ikuko Kobayashi was filed on August 13, 2004 and published on December 22, 2005.[1] Kobayashi is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

To the extent Plaintiff alleges Kobayashi does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Kobayashi or combine the teachings of Kobayashi with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	**Prior Art**
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Kobayashi discloses "[a] method for providing data to a plurality of compute nodes." For example: Kobayashi at claim 3 An information storing method for a storage device in a computer system that comprises a plurality of computers and a shared information storage device shared by the plurality of computers,
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Kobayashi discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: Kobayashi at claim 3 the boot information storage area stores boot information used when booting a computer, wherein said boot information is stored on one computer among said plurality of computers
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	Kobayashi discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." For example: Kobayashi at [0125] According to the computer system 100 of the embodiment, information stored on the local disk of a computer 10 can be readily stored without duplication in a storage device 50a, 50b. Specifically, in the embodiment, unique information is allocated to each file making up the boot information and AP information stored in the boot LU and AP_LU. Unique information is also allocated to each file making up the boot information and AP information stored on the local disk of a new computer 10. Accordingly, by comparing the two, a file not stored can readily be determined. Kobayashi at [0132]

Claims	Prior Art
	Where diskless computers are used, no process for migrating information stored on local disks will be carried out; however, a diskless computer can be network(4) booted by means of having the installer computer 40 store boot information and AP information on storage devices 50a, 50b. For diskless computers making up an already configured computer system, by handling storage devices functioning as storage disks on the network of diskless computers as local disks, it is possible to carry out the information migrating process in the embodiment hereinabove. In this case, the storage devices in the existing computer system can be replaced with the storage devices 50a, 50b in the embodiment. Kobayashi at claim 3 a personal storage area for storing personal information in said storage device, wherein the personal information is unique to an operating environment of each computer and is stored on said one computer Kobayashi at claim 3 storing information that is stored in each said computer and is not currently stored in said boot information storage area in said personal storage area Kobayashi at claim 3 storing said personal information in said formatted personal storage area Kobayashi at claim 11 The information storing method according to claim 10, wherein said computer system comprises a plurality of storage devices; wherein said information storing method further comprises the step of associating one storage device among said plurality of storage devices with one computer among said plurality of computers;
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Kobayashi discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: Kobayashi at [0128] According to the computer system 100 of the embodiment, since access frequency to a given logical unit in a storage device 50a, 50b is high, cache hit rate can be improved, and accessibility to the logical unit can be improved.

Claims	**Prior Art**
[11.0] 11. The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	Kobayashi discloses "[t]he method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments." For example: Kobayashi at [0122] Computer 10 is only able to read from the boot LU and AP_LU, but can both read from and write to personal LU s. Under such conditions, various files needed in various processes using an OS or application program are read from the boot LU, AP_LU or personal LU; and information created by various processes is written to personal LU s. Kobayashi at claim 3 reflecting in said disk-image management information the storage status of information in said storage areas

Exhibit 844-17

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 6,453,334 ("Vinson")

Vinson titled "Method and apparatus to allow remotely located computer programs and/or data to be accessed on a local computer in a secure, time-limited manner, with persistent caching" to Jeffrey Vinson et al. was filed on June 16, 1998 and issued on September 17, 2002.[1] Vinson is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

To the extent Plaintiff alleges Vinson does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Vinson or combine the teachings of Vinson with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	Prior Art
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Vinson discloses "[a] method for providing data to a plurality of compute nodes." For example: Vinson at 2:29–31 The present invention provides a method and apparatus for allowing a remotely-located computer program or data to be accessed on a local computer
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Vinson discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: Vinson at 2:19–21 Only those portions of the program or data actually used are downloaded, Vinson at 2:29–31 The present invention provides a method and apparatus for allowing a remotely-located computer program or data to be accessed on a local computer Vinson at 3:29–32 The chunks are downloaded by the FSD and cached, decompressed, and decrypted as needed during the execution of the target program on the client machine. Vinson at 4:22–26 break the file up into chunks and write the chunks to individual files on the runtime server. The initial size of each chunk (hereafter chunk-size) is a parameter to the Decimation Procedure, and can be tuned to increase performance; Vinson at 4:27–34 The actual decimation into chunks is accomplished (step 122) by opening the file, reading a block of chunk-size bytes, encrypting the block, compressing the block, calculating a unique file name for the

Claims	Prior Art
	block from the unique ID of the file and the block number, writing the just processed block to a new file in the specified directory of the runtime server Vinson at 9:1–7 The primary difference between a baseline network file system driver and the present invention is in controlling access to the remote components, and in maintaining a persistent but secure cache. The FSD creates a virtual drive, complete with drive letter, and places each target program that is to be accessed under a top-level directory on that virtual drive.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	Vinson discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." For example: Vinson at 2:19–21 Only those portions of the program or data actually used are downloaded, Vinson at 3:25–29 The index is used by the client-side file system driver (hereafter the FSD) to present a virtual directory structure to the client machine, including all file properties (file size, last modification time, etc.) and the overall directory structure. Vinson at 9:1–7 The primary difference between a baseline network file system driver and the present invention is in controlling access to the remote components, and in maintaining a persistent but secure cache. The FSD creates a virtual drive, complete with drive letter, and places each target program that is to be accessed under a top-level directory on that virtual drive. Vinson at 13:60–63 Thus, in general, the search for a request block begins with the in-memory cache. If it isn't in the in-memory cache, the on-disk cache is searched. If it isn't in the on-disk cache, the data is downloaded from the runtime server.

Claims	Prior Art
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Vinson discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: Vinson at 2:34–36 A persistent caching scheme improves performance of subsequent accesses. Vinson at 9:1–7 The primary difference between a baseline network file system driver and the present invention is in controlling access to the remote components, and in maintaining a persistent but secure cache. The FSD creates a virtual drive, complete with drive letter, and places each target program that is to be accessed under a top-level directory on that virtual drive. Vinson at 13:28–29 the block size used by VCACHE and the on-disk cache is 4K Vinson at 13:32–34 The on-disk cache block numbers just allocated will be used to contain the data downloaded from the runtime server. Vinson at 13:60–63 Thus, in general, the search for a request block begins with the in-memory cache. If it isn't in the in-memory cache, the on-disk cache is searched. If it isn't in the on-disk cache, the data is downloaded from the runtime server.
[11.0] 11. The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	Vinson discloses "[t]he method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments." For example: Vinson at 9:1–7 The primary difference between a baseline network file system driver and the present invention is in controlling access to the remote components, and in maintaining a persistent but secure cache. The FSD creates a virtual drive, complete with drive letter, and places each target program that is to be accessed under a top-level directory on that virtual drive.

Claims	Prior Art
	Vinson at 13:60–63 Thus, in general, the search for a request block begins with the in-memory cache. If it isn't in the in-memory cache, the on-disk cache is searched. If it isn't in the on-disk cache, the data is downloaded from the runtime server.

Exhibit 844-18

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 5,930,513 ("Taylor")

Taylor titled "Reference based software installation" to Julian S. Taylor was filed on June 6, 1996 and issued on July 27, 1999.[1] Taylor is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

To the extent Plaintiff alleges Taylor does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Taylor or combine the teachings of Taylor with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	Prior Art
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Taylor discloses "[a] method for providing data to a plurality of compute nodes." For example: Taylor at 1:7–11 This invention relates to improvements in computer systems and methods, and more particularly to improvements in computer systems and methods for interfacing server and client computers and for installing computer software and software applications. Taylor at 8:10–11 As shown in FIG. 2, the server computer 20 serves a number of clients
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Taylor discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: Taylor at Abstract A file system for use in a network computer system has a server file system in the server, which contains at least some file elements to be selectively accessed by the client. Taylor at 4:66–5:2 The file system has a server providing a parent file system, which contains at least some file elements to be selectively accessed by the client computer.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the	Taylor discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." For example: Taylor at Abstract

Claims	Prior Art
blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	An overlay file system is assigned to the client within the server Taylor at 5:24–39 In one embodiment, the overlay file system provides an overlay cache to each of the clients. The overlay cache contains only files that are different in the client from those in the parent file system on the server. Moreover, the various external clients are enabled to write files or other information to the overlay cache. The existence of the overlay cache is transparent to the various clients. The overlay cache appears to the clients as if it were in fact the files of the server. Each of the clients, however, is allowed to write information to the overlay cache, and since the overlay cache appears to the clients to be actually files of the server, which indeed it is not, the clients have the illusion of writing to the files of the server. This is particularly useful since the client is enabled to make virtually whatever modifications to the software as the client may desire; however, the actual server files are never modified.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Taylor discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: Taylor at 4:38–41 The data on the server is "cached" on the client significantly reducing network traffic and server load. Even though the data is available from the server, the next time the client reads that data, it will read it from its own local disk. Taylor at 6:46–58 According to the method, an overlay file system is established. The overlay file system identifies file elements that have been installed to a memory, for example, a hard disk, of the computer. A request for access to a file element is directed through the overlay file system, and if the overlay file system indicates that the file element to which access has been requested has been installed in the memory of the computer, the file element is accessed from the memory of the computer. On the other hand, if the overlay file system indicates that the file element to which access has been requested has not been installed in the memory of the computer, the file element is accessed from the software source.
[11.0] 11. The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create	Taylor discloses "[t]he method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments." For example:

Claims	**Prior Art**
cohesive respective application environments.	Taylor at 6:46–58 According to the method, an overlay file system is established. The overlay file system identifies file elements that have been installed to a memory, for example, a hard disk, of the computer. A request for access to a file element is directed through the overlay file system, and if the overlay file system indicates that the file element to which access has been requested has been installed in the memory of the computer, the file element is accessed from the memory of the computer. On the other hand, if the overlay file system indicates that the file element to which access has been requested has not been installed in the memory of the computer, the file element is accessed from the software source. Taylor at 8:19–24 The various applications 34 provided by the server 20 are selectively shared by each of the clients 22 and 24 "through" their respective overlay file systems. Thus, the overlay file system 28 is configured to provide, together with the individual overlay caches 30 and 32, the complete client file system.

Exhibit 844-19

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2002/0002660 ("Malcolm")

Malcolm titled "Multiple cache communication" to Michael A. Malcolm was filed on July 31, 1998 and published on January 3, 2002.[1] Malcolm is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

To the extent Plaintiff alleges Malcolm does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Malcolm or combine the teachings of Malcolm with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	**Prior Art**
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Malcolm discloses "[a] method for providing data to a plurality of compute nodes." For example: Malcolm at [0003]–[0004] 2. Related Art In a computer system in which client devices request information from one or more server devices, it is sometimes desirable to provide a cache; that is, a device that maintains copies of requested information so multiple requests for the same information can be satisfied at the cache. When requests for information are satisfied at the cache, the server devices need not receive the requests, process them, and retransmit the same information over a communication channel that links the client devices and the server devices.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Malcolm discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: Malcolm at [0003]–[0004] 2. Related Art In a computer system in which client devices request information from one or more server devices, it is sometimes desirable to provide a cache; that is, a device that maintains copies of requested information so multiple requests for the same information can be satisfied at the cache. When requests for information are satisfied at the cache, the server devices need not receive the requests, process them, and retransmit the same information over a communication channel that links the client devices and the server devices. Malcolm at [0044] In a preferred embodiment, the graph structure is a tree structure, with a single root cache 111 and a plurality of leaf caches 111. For example, in a cache system 110 disposed for use with an ISP (internet service provider), there is one root cache 111 coupled to an internet backbone, and there is one leaf cache 111 for each POP (point of presence). In this example, the inter-cache communication path 112 includes direct connections (such as T1 or T3 connections) between the root cache 111 and each leaf cache 111. Malcolm at [0051]–[0053]

Claims	Prior Art
	If a client device 120 requests a cacheable web object 133, the leaf cache 111 might already have the requested web object 133 in its memory or storage 112. If so, the leaf cache 111 serves the requested web object 133 to the client device 120 without having to request the web object 133 from the root cache 111 or from the server device 130. If the leaf cache 111 does not already have the requested web object 133, the leaf cache 111 requests it from the root cache 111. The root cache 111 performs a similar caching function, returning the requested cacheable web object 133 directly to the leaf cache 111 if it is already present in its own memory or storage 112, without having to request that web object 133 from the server device 130. If the root cache 111 does not already have the requested web object 133 in its memory or storage 112, the root cache 111 requests it from the server device 120. If the leaf cache 111 and the root cache 111 do not already have a copy of the web object 133 in their respective memory or storage 112, the root cache 111 requests the web object 133 from the server device 120. Similarly, if the web object 133 is considered not cacheable, the root cache 111 requests the web object 133 from the server device 120 whether or not it has already that web object 133 in their respective memory or storage 112. The server device 120 receives the request and returns the requested web object 133 to the root cache 111.

Exhibit 844-20

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent Application Publication No. 2002/0091763 ("Shah")

Shah titled "Client-side performance optimization system for streamed applications" to Lacky Vasant Shah was filed on May 15, 2001 and published on July 11, 2002.[1] Shah is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

To the extent Plaintiff alleges Shah does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Shah or combine the teachings of Shah with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	Prior Art
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Shah discloses "[a] method for providing data to a plurality of compute nodes." For example: Shah at [0083] When using the invention, an end-user requests applications that are resident on remote systems to be launched and run on the end-user's local system. The end-user's local system is called the client or client system, e.g., a desktop, laptop, palmtop, or information appliance. A remote system is a called a server or server system and is located within a collection of one or more servers called a server cluster.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Shah discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: Shah at [0083] When using the invention, an end-user requests applications that are resident on remote systems to be launched and run on the end-user's local system. The end-user's local system is called the client or client system, e.g., a desktop, laptop, palmtop, or information appliance. A remote system is a called a server or server system and is located within a collection of one or more servers called a server cluster. Shah at [0205] With respect to FIG. 10, the application streaming software consists of a streaming block driver 1004. This block driver 1004 provides the abstraction of a physical disk to a native file system 1003 already installed on the client operating system 1002. The driver 1004 receives requests for physical block reads and writes, which it satisfies out of a cache on a standard file system 1003 that is backed by a physical disk drive 1006, 1007. Requests that cannot be satisfied by the cache go to the streaming application server 1005, as before.
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only	Shah discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of

Claims	Prior Art
additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." For example: Shah at [0026] A local copy-on-write file system allows some applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients. Shah at [0205] With respect to FIG. 10, the application streaming software consists of a streaming block driver 1004. This block driver 1004 provides the abstraction of a physical disk to a native file system 1003 already installed on the client operating system 1002. The driver 1004 receives requests for physical block reads and writes, which it satisfies out of a cache on a standard file system 1003 that is backed by a physical disk drive 1006, 1007. Requests that cannot be satisfied by the cache go to the streaming application server 1005, as before. Shah at [0218] A local copy-on-write file system allows some applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients. Installations of applications on file servers typically do not allow the installation directories of applications to be written, so additional reconfiguration or rewrites of applications are usually necessary to allow per-user customization of some settings. Shah at [0219] With respect to FIG. 16, the cache 1602 with extensions for supporting local file customization is shown. Each block of data in the cache is marked as "clean" 1604 or "dirty" 1605. Pages marked as dirty have been customized by the client 1609, and cannot be removed from the cache 1602 without losing client customization. Pages marked as clean may be purged from the cache 1602, as they can be retrieved again from the server 1603. The index 1601 indicates which pages are clean and dirty. In FIG. 16, clean pages are white, and dirty pages are shaded. File 11606 contains only clean pages, and thus may be entirely evicted from the cache 1602. File 21607 contains only dirty pages, and cannot be removed at all from the cache 1602. File 31608 contains some clean and some dirty pages 1602. The clean pages of File 31608 may be removed from the cache 1602, while the dirty pages must remain.

Claims	Prior Art
	Shah at claim 6 The process of claim 1, further comprising the step of: wherein said streaming file system is a copy-on-write file system that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients; wherein each block of data in said persistent cache is marked as clean or dirty; wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization; wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server; providing a cache index; wherein said cache index indicates which pages in said persistent cache are clean and dirty; and wherein said copy-on-write file system references said cache index to determine if a page is clean or dirty.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Shah discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: Shah at [0018] One approach installs an application streaming file system on the client machine that appears to contain the installed application. The application streaming file system receive all requests for code or data that are part of the application and satisfies requests for application code or data by retrieving it from its persistent cache or by retrieving it directly from the streaming application server. Code or data retrieved from the server is placed in the cache for reuse. Shah at [0140] c. Client Cache Manager 207—This component caches the application bits received from the Application Server 107 so that next time a request is made to the same bits, the request can be served by the cache instead of having to go to the Application Server 107. Shah at [0205] With respect to FIG. 10, the application streaming software consists of a streaming block driver 1004. This block driver 1004 provides the abstraction of a physical disk to a native file system 1003 already installed on the client operating system 1002. The driver 1004 receives requests for physical block reads

Claims	Prior Art
	and writes, which it satisfies out of a cache on a standard file system 1003 that is backed by a physical disk drive 1006, 1007. Requests that cannot be satisfied by the cache go to the streaming application server 1005, as before. Shah at claim 6 The process of claim 1, further comprising the step of: wherein said streaming file system is a copy-on-write file system that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients; wherein each block of data in said persistent cache is marked as clean or dirty; wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization; wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server; providing a cache index; wherein said cache index indicates which pages in said persistent cache are clean and dirty; and wherein said copy-on-write file system references said cache index to determine if a page is clean or dirty.
[11.0] 11. The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	Shah discloses "[t]he method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments." For example: Shah at [0205] With respect to FIG. 10, the application streaming software consists of a streaming block driver 1004. This block driver 1004 provides the abstraction of a physical disk to a native file system 1003 already installed on the client operating system 1002. The driver 1004 receives requests for physical block reads and writes, which it satisfies out of a cache on a standard file system 1003 that is backed by a physical disk drive 1006, 1007. Requests that cannot be satisfied by the cache go to the streaming application server 1005, as before. Shah at claim 25 A process for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising the steps of:

Claims	Prior Art
	providing a streaming block driver on said client; wherein said block driver provides the abstraction of a physical disk to a native file system already installed on the client operating system; providing a persistent cache on said client; wherein said block driver receives requests for physical block reads and writes from local processes which it satisfies out of said persistent cache on a standard file system that is backed by a physical disk drive; and wherein requests that cannot be satisfied by said persistent cache are sent to said server.

Exhibit 844-21

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 7,398,382 ("Rothman")

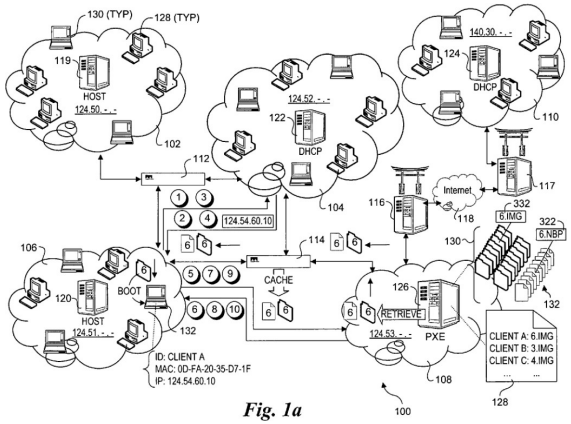
Rothman titled "Method and apparatus to enhance platform boot efficiency" to Michael A. Rothman was filed on December 29, 2004 and issued on July 8, 2008.[1] Rothman is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. § 102(e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

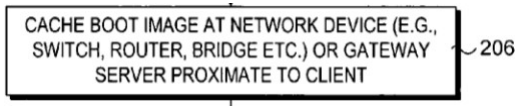
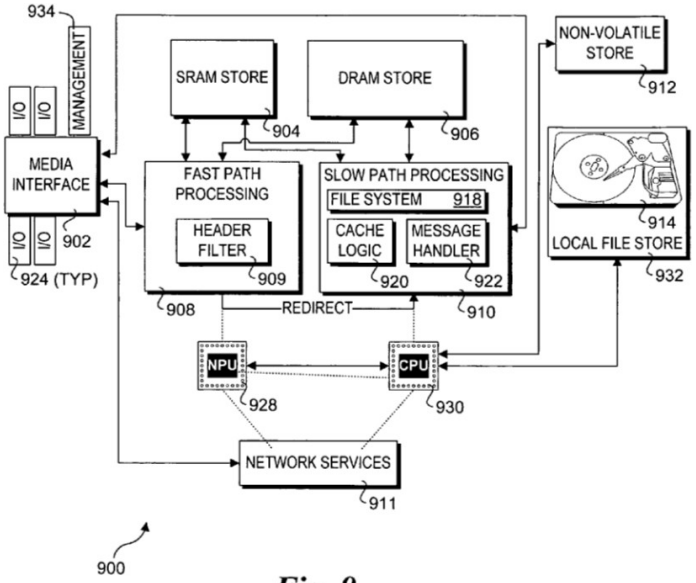
To the extent Plaintiff alleges Rothman does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Rothman or combine the teachings of Rothman with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	**Prior Art**
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Rothman discloses "[a] method for providing data to a plurality of compute nodes." For example: Rothman at Abstract Methods and apparatus for improving network boot efficiency are disclosed. Under embodiments of the method, boot images that are initially sent from a boot server to various clients are cached at network devices along communication paths between the boot server and the clients.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Rothman discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: Rothman at Abstract Methods and apparatus for improving network boot efficiency are disclosed. Under embodiments of the method, boot images that are initially sent from a boot server to various clients are cached at network devices along communication paths between the boot server and the clients. Rothman at Figure 1a.  *Fig. 1a*

Claims	Prior Art
	Rothman at Fig. 2  Rothman at Fig. 9  ***Fig. 9*** Rothman at 2:60–61 Embodiments of methods and apparatus for improving network boot efficiency are described herein. Rothman at 4:6–11 In a block 204, the client obtains an operating system boot image over the network from a boot server. In conjunction with delivering the boot image to the client, the boot image is cached at a network device, such as a switch, router, bridge, etc., or a gateway server that is proximate to the client in a block 206. Rothman at 11:1–6 In general, a cached filed will be stored on a local non-volatile store, such as a local hard disk built into the network device chassis or otherwise communicatively-coupled to the network device. For example,

Claims	Prior Art
	the network device could actually store the cached files on a local file server that it accesses via a network link. Rothman at 14:16–25 Architecture blocks corresponding to an exemplary network device 900 that supports the network device caching operations described above are shown in FIG. 9. The architecture blocks include a media interface 902, an SRAM store 904, a DRAM store 906, a fast path processing block 908, a slow path processing block 910, network services 911, a non-volatile (NV) store 912, and a local file store 914. The fast path processing block includes a packet header filter 916, while the slow path processing block includes a file system 918, cache management logic 920, and message handler 922. Rothman at 15:32–39 A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium can include such as a read only memory (ROM); a random access memory (RAM); a magnetic disk storage media; an optical storage media; and a flash memory device, etc.

Exhibit 844-22

Invalidity of U.S. Patent No. 8,332,844 in View of United States Patent No. 6,317,826 ("McCall")

McCall titled "Booting a computer system from a network" to Colin David McCall was filed on October 19, 1998 and issued on November 13, 2001.[1] McCall is prior art to the '844 Patent under at least Pre-AIA 35 U.S.C. §§ 102(b) and (e) as to claims 7 and 11 (the "Asserted Claims") of the '844 Patent, as discussed in detail in the chart that follows.

To the extent Plaintiff alleges McCall does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify McCall or combine the teachings of McCall with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, such as the other prior art identified in Defendant's invalidity contentions.

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 421.

Defendant reserves the right to amend or supplement this claim chart at a later date as more fully set forth in the Invalidity Contentions.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Claims	**Prior Art**
[7.0] 7. A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, McCall discloses "[a] method for providing data to a plurality of compute nodes." For example: McCall at Abstract A client computer system (10) is adapted to connect to a server (16) across a network (20). The client system includes memory (25 or a storage medium) into which an operating system is loadable. The client includes a network card (12) having a PROM (14) adapted to download a bootstrap program (17) from the server. The bootstrap program (17) is in turn adapted to load an operating system image (18) over the network from the server into a location in the memory.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	McCall discloses "storing blocks of a root image of said compute nodes on a first storage unit." For example: McCall at Abstract A client computer system (10) is adapted to connect to a server (16) across a network (20). The client system includes memory (25 or a storage medium) into which an operating system is loadable. The client includes a network card (12) having a PROM (14) adapted to download a bootstrap program (17) from the server. The bootstrap program (17) is in turn adapted to load an operating system image (18) over the network from the server into a location in the memory.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	McCall discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." For example: McCall at Abstract The client then boots from the operating system image in memory, with the client system being adapted to subsequently modify the operating system image in memory, without reloading the image over the network, and re-boot the client system. McCall at 5:27–38

Claims	Prior Art
	A client computer system adapted to connect to a server across a network, the client system including memory into which an operating system is loadable and being adapted to load at least a portion of an operating system image over the network from the server into a location in said memory and to boot from the operating system image in memory, wherein said client system is adapted to subsequently modify the operating system image in memory without reloading the portion of the operating system image over the network and to re-boot the client system from the modified operating system image in memory also without reloading the portion of the operating system image over the network.

Exhibit 844-23 to Southwest's Preliminary Invalidity Contentions

Southwest Airlines NetApp System

Southwest Airlines NetApp System was known in this country, used in public, sold or offered for sale no later than 2002, as demonstrated by at least the materials cited herein or the testimony of knowledgeable witnesses and corroborating documents.[1] Specifically, based on information currently available, Southwest Airlines NetApp System was (1) known or used in this country before the alleged invention of the claimed subject matter of the asserted claims, (2) in public use or on sale in this country more than one year before the filing date of the patent, or (3) invented by another who did not abandon, suppress, or conceal, before the alleged invention of the claimed subject matter of the asserted claim. Thus, Southwest Airlines NetApp System anticipates the asserted claims under at least 35 U.S.C. §§ 102(a) or (g).

At least the following documents describe the relevant functionality disclosed by Southwest Airlines NetApp System:

- https://web.archive.org/web/20040708152749/http://www.netapp.com/news/feature/2002/southwest.pdf;
- https://web.archive.org/web/20040719184445/http://www.netapp.com:80/ftp/snapshot-brochure.pdf;
- https://web.archive.org/web/20040805003209/http:/www.netapp.com/ftp/reliability-brochure.pdf;
- https://web.archive.org/web/20040717035040/http:/www.netapp.com/ftp/product-overview.pdf;
- https://web.archive.org/web/20040719153814/http://www.netapp.com:80/ftp/snapshot-brief.pdf; and
- https://web.archive.org/web/20030322200556/http://www.netapp.com:80/ftp/snaprestore.pdf.

Based on information currently available, the persons or entities involved in the conception and diligent reduction to practice of the ideas in Southwest Airlines NetApp System include at least persons involved in the Southwest Airlines NetApp System and their testing and commercial use, including at least Southwest Airlines Co. The conception of Southwest Airlines NetApp System occurred at least as early as 2002.

Southwest offers this invalidity chart based on its current understanding of Southwest Airlines NetApp System. Southwest's investigation of this system is ongoing. Southwest may rely on materials produced by Plaintiffs or by third parties regarding Southwest Airlines NetApp System and related software to demonstrate invalidity of the asserted claims. Southwest has not had an opportunity to review source code for Southwest

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

Airlines NetApp System. Southwest may rely on source code, when available, implementing the functionality described herein, or other source code implementing substantially similar functionality, to demonstrate the invalidity of the asserted claims.

To the extent Plaintiff alleges Southwest Airlines NetApp System does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify Southwest Airlines NetApp System or combine the teachings of Southwest Airlines NetApp System with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc*., "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id*. at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id*. at 421.

Claims	Southwest Airlines NetApp System
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, Southwest Airlines NetApp System discloses: The NetApp system implemented at Southwest Airlines in 2002 exemplifies a robust and efficient

	method for managing root and leaf image data. The system architecture at Southwest leveraged clustered NetApp filers integrated with Oracle9i RAC to achieve high availability and seamless data management. The key technologies used in this setup included WAFL (Write Anywhere File Layout), SnapMirror, and SnapRestore. These technologies facilitated the management of root and leaf image data, maintained consistent application environments, and supported rapid recovery after system disruptions.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	Southwest Airlines NetApp System discloses "storing blocks of a root image of said compute nodes on a first storage unit." *See, e.g.*, The root image, which contains the foundational system data and configuration, was maintained on a centralized NetApp Filer within the Southwest Airlines system. WAFL technology, a core component of the NetApp system, enabled efficient storage and handling of snapshots by managing data without duplicating unchanged blocks. This approach ensured that the root image could be consistently accessed and maintained across multiple compute nodes. The root image is stored on a NetApp Filer, utilizing WAFL (Write Anywhere File Layout) for efficient data management. The root image is maintained as a snapshot. As highlighted in the Southwest Airlines case study, "[Southwest Airlines] switched from a single system to a clustered approach based on Oracle9i ™ Real Application Clusters (RAC), which makes it possible to run an Oracle® database across a cluster of multiple servers. If one server in a cluster goes down, another can pick up the workload instantly, and the system can be easily expanded by adding more servers. . . . NetApp filers are able to handle storage tasks with speed and efficiency, and they feature storage-specific intelligence and tools. . . . 'The tight integration between Oracle9i RAC and NetApp filers gives [Southwest Airlines] the highly available total package we need for these key applications.'" https://web.archive.org/web/20040708152749/http://www.netapp.com/news/feature/2002/southwest.pdf

[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	Southwest Airlines NetApp System discloses "storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes." *See, e.g.*, When data modifications occurred, WAFL's snapshot technology used Copy-on-Write (CoW) to efficiently record changes without duplicating the entire root image. Instead, WAFL captured only the data blocks that had been altered, creating leaf images that contained only additional data not previously present in the root image. This method significantly conserved storage space while maintaining the integrity of the original root image. NetApp documentation notes that "A Snapshot copy is a 'frozen,' read-only view of a WAFL volume that provides easy access to old versions of files, directory hierarchies, and/or LUNs (logical unit numbers). The system creates a Snapshot copy by preserving all the currently used disk blocks. A Snapshot copy is typically created in less than one second, regardless of the size of the volume or the level of activity on the NetApp storage system. After a Snapshot copy has been taken, changes to data objects are reflected in updates to the current version of the objects, as if the Snapshot copy did not exist. Meanwhile, the Snapshot versions of the data remain completely stable." https://web.archive.org/web/20040719184445/http://www.netapp.com:80/ftp/snapshot-brochure.pdf "A core capability of NetApp technology, Snapshot software delivers near-instantaneous, transparent, online incremental images of your data. It requires minimal disk space and operates invisibly in the background." https://web.archive.org/web/20040805003209/http:/www.netapp.com/ftp/reliability-brochure.pdf Snapshots in WAFL capture only the changes since the last snapshot, keeping the base root image intact. Leaf images contain only additional data blocks and changes made by the compute nodes. The snapshot process in WAFL ensures that unchanged data remains part of the root image, while only the modified data is stored separately as a leaf image. This efficient use of space aligns with the patent claim requirement that the leaf images exclude unchanged data. According to NetApp

	documentation, "Snapshot copies write only changed 4kB blocks to the active file system. Even the most efficient competitive implementations consume twice the storage volume of the active file system." https://web.archive.org/web/20040719184445/http://www.netapp.com:80/ftp/snapshot-brochure.pdf "A core capability of NetApp technology, Snapshot software delivers near-instantaneous, transparent, online incremental images of your data. It requires minimal disk space and operates invisibly in the background." https://web.archive.org/web/20040805003209/http:/www.netapp.com/ftp/reliability-brochure.pdf "[Snapshot] Enables online backups, providing near-instantaneous access to previous versions of data without requiring complete, separate copies." https://web.archive.org/web/20040717035040/http:/www.netapp.com/ftp/product-overview.pdf WAFL's CoW-based snapshot technology ensures that only changes made by the compute nodes are stored as leaf images, while unchanged data remains part of the root image. *"After a Snapshot copy has been created, changes to data objects are reflected in updates to the current version of the objects, as if Snapshot copies did not exist. Meanwhile, the Snapshot versions of the data remain completely stable. Therefore, Snapshot copies incur no performance overhead; users can comfortably store up to 255 Snapshot copies per WAFL volume, all of which are accessible as read-only and online versions of the data."* https://web.archive.org/web/20040719153814/http://www.netapp.com:80/ftp/snapshot-brief.pdf To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique.

	Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5. "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.

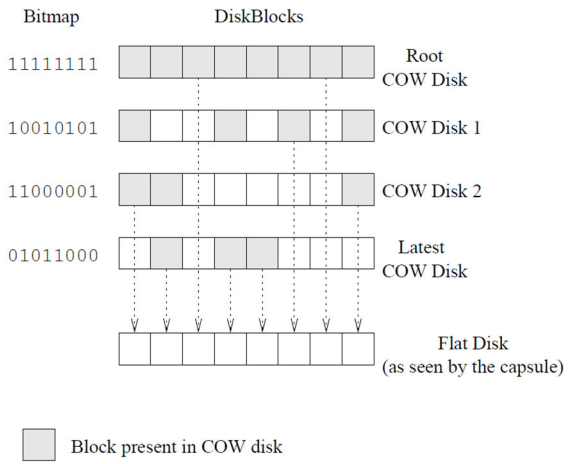


Figure 2: An example capsule disk.

Sapuntzakis at 6.

See also Sapuntzakis at 1-12.

> "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally."

Birse at 7:35-53; *see also* 11:30-32.

	"In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139." Kulkarni at [0037]. "The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038].

	See also Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Southwest Airlines NetApp System discloses "caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory." *See, e.g.*, To enhance performance and reduce latency, the Southwest system utilized Data ONTAP's caching features. Frequently accessed root image data was temporarily stored in cache memory, which minimized disk I/O and facilitated faster data retrieval. This capability was crucial for maintaining high performance during peak system demands. As explained in the NetApp reliability brochure, "Our Data ONTAP™ microkernel ensures reliable operation through high-performance, highly optimized code." https://web.archive.org/web/20040805003209/http://www.netapp.com/ftp/reliability-brochure.pdf "[Data ONTAP is] Operating system software that optimizes data serving by combining patented file system technology and a microkernel design dedicated to multiprotocol data access." https://web.archive.org/web/20040717035040/http:/www.netapp.com/ftp/product-overview.pdf Data ONTAP employs in-memory caching to optimize data retrieval and minimize disk I/O. *"Southwest Airlines discovered the value of Snapshot technologies early on, says Kerry Schwab. During implementation, IT professionals could take a Snapshot at key points, and then try a number of options. 'If we decided against going down a certain path, we could start over very quickly,' he says. 'We could run through some what-if scenarios—what if I increase parallelism on this, or what if I spread the data out this way instead of that way? If we had been using traditional storage, we would have had to take several hours to make a backup, and several more hours to restore. The Snapshot feature let us try things without the incremental costs of dumping and restoring all the data.'"* https://web.archive.org/web/20040708152749/http://www.netapp.com/news/feature/2002/southwest.pdf

	To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

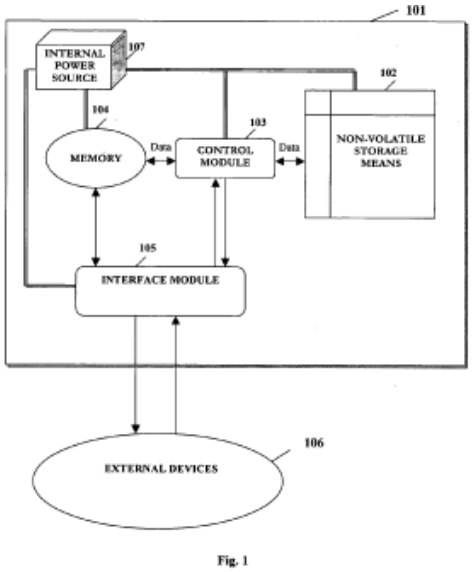


Fig. 1

Holzmann at FIG. 1

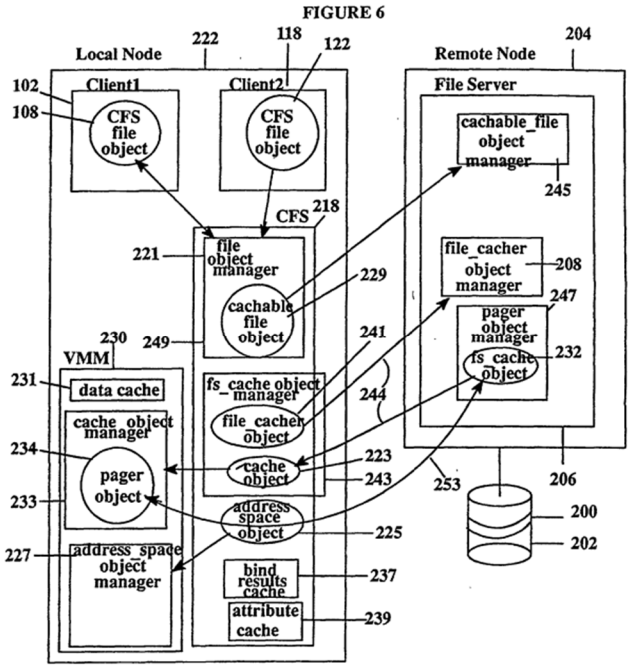
> "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107."

Holzmann at [0017].

> "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to

	the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].



Nelson at FIG. 6.

See also Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4.

> "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for

	example, information about what files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	Southwest Airlines NetApp System discloses "merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments." *See, e.g.*, An essential feature of the Southwest system was its ability to rapidly restore a consistent application environment after data loss or system changes. NetApp's SnapRestore technology played a pivotal role in this process by merging the root image with the modifications stored in snapshots. This rapid restoration ensured that the system could quickly revert to a previous stable

	state. The documentation emphasizes that "[SnapRestore] Allows rapid restoration of a file system to an earlier point in time, typically in only a few seconds." https://web.archive.org/web/20040717035040/http:/www.netapp.com/ftp/product-overview.pdf SnapRestore technology from NetApp merges the root image with the changes made by compute nodes, creating a cohesive application environment after recovery. "SnapRestore software uses Snapshot technology to perform near-instantaneous data restoration. . . . With SnapRestore, data can be restored from any one of the Snapshots stored on the file system. This allows an application development team, for example, to revert to Snapshots from various stages of their design, or test engineers to quickly and easily return data to a baseline state. Restoring to the base environment takes only seconds, and the restored environment is identical to the point at which the Snapshot copy was created." https://web.archive.org/web/20030322200556/http://www.netapp.com:80/ftp/snaprestore.pdf "While Snapshot software is perfect for recovering single files, SnapRestore is made for recovering multiterabyte volumes. SnapRestore uses Snapshot technology to perform near-instantaneous data restoration at the volume level, recovering an entire file system back to a known good state in seconds without interrupting users and applications on other volumes." https://web.archive.org/web/20040805003209/http:/www.netapp.com/ftp/reliability-brochure.pdf To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*,

	"To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience." Birse at 9:2-10.

	"Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013]. "The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the

	new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.

Exhibit 844-24 to Southwest's Preliminary Invalidity Contentions

VMware Systems

VMware Systems was known in this country, used in public, sold or offered for sale at least as early as 2001, as demonstrated by at least the materials cited herein or the testimony of knowledgeable witnesses and corroborating documents.[1] Specifically, based on information currently available, VMware Systems was (1) known or used in this country before the alleged invention of the claimed subject matter of the asserted claims, (2) in public use or on sale in this country more than one year before the filing date of the patent, or (3) invented by another who did not abandon, suppress, or conceal, before the alleged invention of the claimed subject matter of the asserted claim. Thus, VMware Systems anticipates the asserted claims under at least 35 U.S.C. §§ 102(a) or (g).

At least the following documents describe the relevant functionality disclosed by VMware Systems:

- VMware GSX Server 3 Virtual Machine Guide;
- VMware GSX Server 3 Specifications Datasheet; and
- VMware ESX Server 2.1 Specifications Datasheet.

Based on information currently available, the persons or entities involved in the conception and diligent reduction to practice of the ideas in VMware Systems include at least persons involved in the VMware Systems and their testing and commercial use, including at least VMware. The conception of VMware Systems occurred at least as early as 2001.

Southwest offers this invalidity chart based on its current understanding of VMware Systems. Southwest's investigation of this system is ongoing. Southwest may rely on materials produced by Plaintiffs or by third parties regarding VMware Systems and related software to demonstrate invalidity of the asserted claims. Southwest has not had an opportunity to review source code for VMware Systems. Southwest may rely on source code, when available, implementing the functionality described herein, or other source code implementing substantially similar functionality, to demonstrate the invalidity of the asserted claims.

[1] Defendant's invalidity charts, in some instances, rely at least in part on Plaintiff's apparent positions regarding the scope of its claims for purposes of asserting infringement. Nothing in these claim charts should be understood as an admission relating to infringement, either literally or under the doctrine of equivalents, or as an admission relating to Defendant's understanding of the proper interpretation or scope of the Asserted Claims. Defendant reserves the right to rely on additional citations or sources of evidence that may also be applicable, or that may become applicable in light of any Court Order on claim construction, changes in Plaintiff's infringement contentions, or information obtained during discovery as the case proceeds.

To the extent Plaintiff alleges VMware Systems does not disclose any particular limitation of the Asserted Claims of the '844 Patent, either expressly or inherently, any purported differences are such that the claimed subject matter as a whole would have been obvious in view of the knowledge of one skilled in the art. It would have further been obvious to a person of ordinary skill in the art as of the priority date of the '844 Patent to modify VMware Systems or combine the teachings of VMware Systems with other prior art in a manner that would have rendered the Asserted Claims invalid as obvious, including but not limited to:

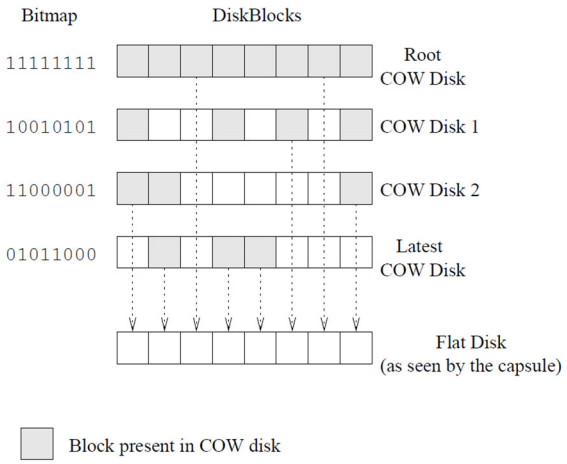
- U.S. Patent Application Publication No. 2005/0283597 ("Holzmann");
- EP Patent No. EP0604013 ("Nelson");
- U.S. Patent Application Publication No. 2005/0004925 ("Stahl");
- Optimizing the Migration of Virtual Computers" ("Sapuntzakis");
- U.S. Patent No. 7,089,300 ("Birse"); and/or
- U.S. Patent Application Publication No. 2002/0034105 ("Kulkarni").

As the United States Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc*., "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held: "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id*. at 417. The Supreme Court further held: "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id*. at 421.

Claims	VMware Systems
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, VMware Systems discloses: VMware GSX Server was introduced in January 2001, and VMware ESX Server was introduced in March 2001. These systems enable multiple virtual machines (compute nodes) to run concurrently on a single physical server. The virtual machines share data through root image files stored on centralized storage units and manage data changes via snapshots and redo logs. The

	systems efficiently provide data to a plurality of compute nodes through virtualization and dynamic resource allocation.
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	VMware Systems disclose “storing blocks of a root image of said compute nodes on a first storage unit.” *See, e.g.*, The VMware GSX Server stores virtual machine configurations (including root images) in a dedicated directory, consistent with storing root image blocks on a first storage unit. "The virtual machine typically is stored on the host computer in a set of files, all of which are in a directory set aside for that particular virtual machine." VMware GSX Server 3 Virtual Machine Guide VMware GSX Server stores virtual machine data, including root images, as encapsulated virtual disk files on a storage unit. "Virtual disk files encapsulate all virtual machine data and can run on any system with VMware GSX Server, ESX Server, or Workstation installed." VMware GSX Server 3 Specifications Datasheet VMware ESX Server stores root images as isolated entities within a physical system, effectively using a storage unit for each VM’s root image. "VMware ESX Server transforms physical systems into a pool of logical computing resources... Operating systems and applications are isolated in multiple virtual machines that reside on a single piece of hardware." VMware ESX Server 2.1 Specifications Datasheet
[7.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously	VMware Systems disclose “storing leaf images for respective compute nodes on respective second storage units, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said

contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	root image that are unchanged by respective compute nodes." *See, e.g.*, After a snapshot is taken, changes are stored in redo-log files, which act as leaf images containing only the modified data blocks. "Once you take a snapshot, the virtual machine starts saving any changes to the virtual machine to one or more redo-log files." VMware GSX Server 3 Virtual Machine Guide The snapshot mechanism captures only the changes made during runtime, storing modified data separately from the unchanged root image. "Snapshots save a point-in-time copy of the running virtual machine state." VMware ESX Server 2.1 Specifications Datasheet To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, storing data blocks not previously contained in a root image and/or changes made by compute nodes to the blocks of the root image was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Sapuntzakis at 5.

	"Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Sapuntzakis at 5–6. "Since the latest COW disk is always local, all writes are local." Sapuntzakis at 6.  Figure 2: An example capsule disk. Sapuntzakis at 6. *See also* Sapuntzakis at 1-12. "The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver. However, in reality, the block device driver 154 may be configured to service the read and write requests of the file system 152 by accessing

	the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes, may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." Birse at 7:35-53; *see also* 11:30-32. "In the embodiment depicted, it is assumed that the shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified. The various options regarding the granularity of the portions written to the shadow volume are discussed below." Birse 11:49-52. *See also* Birse at 10:63-65, 11:6-17, 12:54–59, 14:49-61, FIGS. 4, 6, 10. "FIG. 2 is a flow diagram outlining the acts performed by the server 102 to create the differences file 120 from the old image 114 and the new image 116. In the present example, the updated image is an updated device operating system for the PDA 104." Kulkarni at [0036]. "Creating the differences file 120 is accomplished by applying a compression algorithm to the new image 116, using the old image 114 as a reference. Since the old image 114 is already stored on the PDA 104, data from the new image 116 that is also contained in the old image 114 does not need to be downloaded from the server 102 to the PDA 104. When the new image 116 is created at the PDA 104, data that is contained in both images is copies from the image 139 already on the PDA 104. Data that is only contained in the new image 116 is downloaded to the PDA 104 in the modified differences file 124 and is copied to the flash memory image 139."

	Kulkarni at [0037]. "The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header." Kulkarni at [0038]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Holzmann, Nelson, and/or Stahl. Holzmann, Nelson, and Stahl concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known and commonly-used technique. Modifying this reference with Holzmann's, Nelson's, and/or Stahl's caching teachings would have only required simple substitution, would not have required extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Holzmann, Nelson, or Stahl for example, to incorporate this limitation. *See, e.g.*, "The invention is directed generally to computer storage systems. More specifically, the invention is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Holzmann at [0002].

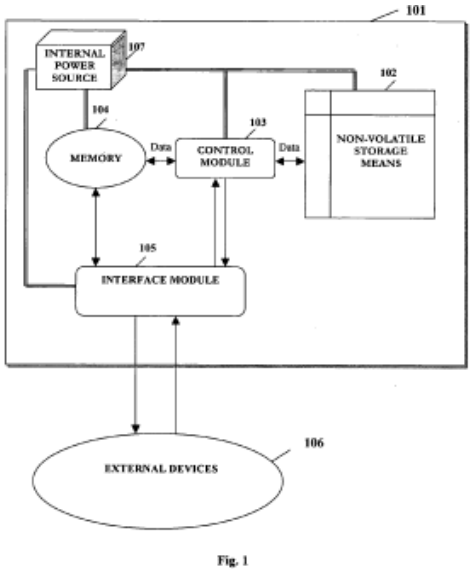


Fig. 1

Holzmann at FIG. 1

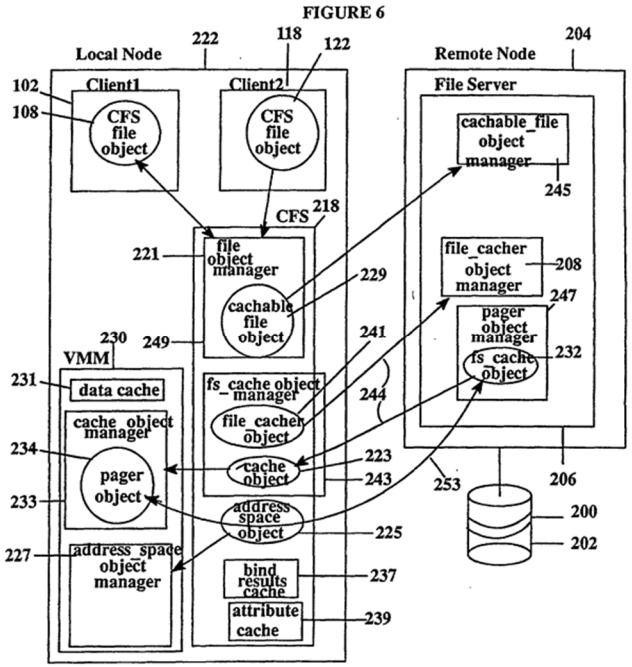
> "Referring now to the figures, the present invention is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107."

Holzmann at [0017].

> "The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module."

	Holzmann at [0020]. "This invention relates in general to the fields of Computer Operating Systems, Multi-processor Hardware Systems, Object Oriented Programming, and Virtual Memory Systems. In particular, the invention relates to improved techniques for establishing and efficiently handling relationships between a clientprogram, a file system server and a virtual memory manager ("VMM")." Nelson at [0001]. "This servicing of multiple client programs is now illustrated by means of Figure 6. A first client program 102 asks the CFS 218 to cache remote file 200. The CFS 218 provides a first CFS file object 108 for the first client 102 to use in processing requests on the file. The CFS 218 establishes a communications link 244 to the file server 206 located on the remote node 204 to obtain the file attributes which the CFS 218 caches in the CFS attribute cache 239. The CFS 218 also establishes the communications between the CFS file object manager 221 and the VMM 230 which caches the file data in the data cache 231 by means of the pager_object 234/fs_cache_object 232 pipeline. This pipeline results from the map and bind operations previously described. Pager object 234 is connected to fs_cache object 232 in the pager object manager 247 , and also connected via the link between the fs_cache object 232 and the cache object 223 in the fs_cache object manager 243 in the CFS 218. When a second client program 118 asks the CFS 218 to cache the same remote file 200, the CFS 218 implements a second CFS file object 122. However, since the CFS 218 has already set up a CFS attribute cache 239 to handle file attributes for the remote file 200, the second CFS file object 122 is merely directed to the existing CFS attribute cache 239. Similarly, when the second CFS file object 122 wishes to read or write data to the file 200, the file object manager 221 again invokes an operation to map the file in the CFS address_space using its address_space object 225. Once again this causes the VMM's address_space object manager 227 to invoke a bind ( ) operation on the file object 122 which is again directed to

	the CFS 218. This time the CFS 218 checks its cache of bind results 237 to see if that file 200 is already being cached by the VMM 230. When the CFS 218 finds that it is already being cached, the CFS 218 returns a pointer to the fs_cache_object/pager_object pair being used, and the bind operation is completed. Thus, duplicate caches are not established and the redundant network communications overhead is avoided by the process." Nelson at [0038]-[0039]. "In an alternate embodiment, if the first client program 102 had set up the first CFS file object 108 with the CFS 218, the first client program 102 may pass the first CFS file object 108 to the second client program 118. In that case, both client programs 102 & 118 would use the same CFS file object 108, and as before both client programs would be serviced by the CFS 218 with only one set of caches." Nelson at [0040].



Nelson at FIG. 6.

See also Nelson at [0010], [0034]-[0037], [0052]-[0119], FIG. 4.

> "Repository FS Implementation 222 is an implementation of a file system responsible for maintaining data on the persistent read-only file system 226 that is to have its data shared. Repository FS Implementation 222 receives and responds to file system requests from MapFS Implementation 212. Repository FS Instance in memory 224 represents a transient in-memory status of a particular Repository file system in persistent storage. Status includes, for

	example, information about what files exist (or do not) and have been successfully or unsuccessfully looked up, file size, permission data, actual cached data blocks, etc." Stahl at [0028]. "The mapping module 502 looks 308 in the View FS instance 208 (via file system communication module 506) to see whether the file is cached. Preferably, the view caches filenames in a manner similar to MapFS—that is, in a table of names and look-up results. Again, since this is the first time the file is being looked up, it will not exist in the View's cache either, unless it has been looked up through some non-MapFS process. Thus, the mapping module 502 asks the 310 View FS implementation 206 to look up the name in the View in a manner preferably similar to the way in which the VFS layer examines MapFS instance data described above. The View FS Implementation 206 looks up 312 the file in the persistent view FS instance on disk 210 and updates 314 the View FS instance in memory 208 by populating an in-memory description of the file, and inserting an entry in its name-to-inode table. The View FS implementation 206 returns 316 a reference to the file in the View FS Instance 208 to the mapping module 502." Stahl at [0037]. *See also* Stahl at paragraphs [0017], [0044].
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	To the extent this reference does not expressly or inherently disclose this limitation, it would have been obvious in view of the background knowledge of a person of ordinary skill in the art. This reference further renders the limitation obvious in light of other references, such as Sapuntzakis, Birse, and/or Kulkarni. Sapuntzakis, Birse, and Kulkarni concern similar subject matter as this reference, namely, distributed computing systems. As set forth in the accompanying contentions, merging data blocks for multiple computer nodes in distributed environments was a well-known and commonly-used technique. Modifying this reference with Sapuntzakis, Birse, and/or Kulkarni's teachings would have only required simple substitution, would not have required

	extensive experimentation, and would have yielded predictable results. A person of ordinary skill in the art would have further understood how to modify the disclosures of this reference with those of Sapuntzakis, Birse, and/or Kulkarni for example, to incorporate this limitation. *See, e.g.*, "To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by- page on demand, rather than being pre-fetched." Sapuntzakis at 6. "Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found." Sapuntzakis at 5-6. "Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Sapuntzakis at 6. See also *Sapuntzakis* at 1-12. "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Birse at 14:61-67. "When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience."

	Birse at 9:2-10. "Referring now to FIG. 5, NC client read request processing will now be described according to one embodiment of the present invention. ….At step 510, the file system 152 generates a read request directed to the system volumes 160. At step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged. If the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530. However, if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540. At step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174. At step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume. In this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." Birse at 11:39-41, 11:55-12:10. "The implementations described herein create and utilize a differences file to create a new image in flash memory of an electronic device from sections of an old image in the flash memory and sections of a new image, the new image being an updated version of the old image. The new image is created incrementally in flash memory so that an entire image does not have to first be constructed in device RAM, thereby saving significant overhead. A "flash manager" is described which allows the new image to be built incrementally until a memory block is created that is of sufficient size to be written to the flash memory." Kulkarni at [0013].

	"The flash manager 136 uses the modified differences file 124′ and the image 139 in flash memory 132 (the old image) to start building the new image. At step 404, a portion of the new image is constructed and temporarily stored in the new memory block 138. When the new memory block 138 contains a portion of the new image that constitutes at least one-half of a write block size for the flash memory 132 ("Yes" branch, step 406), then the portion of the new image stored in the new memory block 138 is added to the image 139 in the flash memory 132. As previously discussed, the size of a write block depends on the type of flash memory, i.e., the brand of the flash memory. It is common for flash memory to have a write block of 64 Kb. Also as previously discussed, the flash manager 136 may manage several new memory blocks 138. When any of the new memory blocks 138 reaches a size of one-half of a write block size, that new memory block 138 is written to the flash memory." Kulkarni at [0069]. *See also* Kulkarni at [0026]-[0041], [0068], FIG. 1, FIGS. 2a-2d, FIG. 6.