

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMERICAN AIRLINES, INC. AND SOUTHWEST AIRLINES CO.,

Petitioners

v.

INTELLECTUAL VENTURES II LLC

Patent Owner.

IPR2026-00043

U.S. Patent No. 7,721,282

**DECLARATION OF EREZ ZADOK IN SUPPORT OF
PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,721,282**

Table of Contents

I.	ASSIGNMENT.....	1
II.	BACKGROUND AND QUALIFICATIONS.....	1
III.	MATERIALS CONSIDERED.....	12
IV.	LEVEL OF ORDINARY SKILL IN THE ART.....	14
V.	RELEVANT LEGAL STANDARDS.....	16
VI.	SUMMARY OF OPINIONS.....	20
VII.	TECHNOLOGY OVERVIEW.....	20
	A. General Computer Operations.....	21
	1. Networking Overview.....	22
	B. Data Storage Devices and File Systems.....	27
	1. Files and File Systems	28
	C. Networked and Distributed Storage Systems.....	32
	D. Copy-on-Write and Snapshots	36
	E. Virtualization.....	49
VIII.	THE '282 PATENT.....	53
	A. Overview	53
	B. The '282 Patent's Union Block Device (UBD)	60
	C. Prosecution History	62
	1. The '816 Application	62
	2. The '622 Application	66
	3. Priority Date.....	68

IX.	THE CLAIMS	69
X.	OVERVIEW OF THE PRIMARY PRIOR ART REFERENCES.....	75
A.	Birse.....	75
B.	Ballew.....	84
C.	Wu	87
D.	Kleinschnitz.....	89
XI.	CLAIM CONSTRUCTION	90
XII.	DETAILED EXPLANATION OF GROUNDS.....	91
A.	Ground 1: Birse Renders Claim 1, 3-10, 13-16, 19-26, 29, 30 Obvious.	91
1.	Claim 1	91
a.	[1.0] A system for distributing an application environment comprising:	91
b.	[1.1] a compute node comprising a computer system; a first storage unit for storing blocks of a root image of the compute node, ...	92
c.	[1.2] wherein the first storage unit comprises a first non- volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;	98
d.	[1.3] a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, wherein the second storage unit comprises a second non-volatile memory; and	101
e.	[1.4] a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute	

	node, wherein the union block device comprises a driver, ...	110
f.	[1.5] wherein the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit;	111
g.	[1.6] the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node;	115
h.	[1.7] and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X.	116
2.	Claim 3: The system as recited in claim 1 wherein the compute node comprises a thin-client workstation.	121
3.	Claim 4: The system as recited in claim 1 wherein the root image comprises an operating system.	121
4.	Claim 5: The system as recited in claim 1 wherein the root image is concurrently accessible to a plurality of compute nodes.	122
5.	Claim 6: The system as recited in claim 1 wherein the first storage unit is remotely located from the compute node.....	124
6.	Claim 7: The method as recited in claim 7 wherein the second storage unit is remotely located from the compute node.....	126
7.	Claim 8: The system as recited in claim 1 wherein the second storage unit contains a block modification log for the compute node.	128
8.	Claim 9: The system as recited in claim 1 wherein the first storage unit in [sic] contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.	129
9.	Claim 10: The system as recited in claim 1 wherein the union block device writes sector X on the second storage unit.	131

10.	Claim 13: The system as recited in claim 1 wherein the first storage unit is read only.	132
11.	Claim 14: The system as recited in claim 1 further comprising an intermediate image between the root image and the leaf image, wherein the intermediate image comprises an application environment service group.....	133
12.	Claim 15.....	139
a.	[15.0] A method for distributing an application environment comprising:.....	139
b.	[15.1] storing blocks of a root image of a compute node on a first storage unit, wherein the compute node comprises a computer system,.....	141
c.	[15.2] and wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;.....	141
d.	[15.3] storing a leaf image comprising new data blocks and changes to the blocks of the root image on a second storage unit, wherein the second storage unit comprises a second non-volatile memory;	141
e.	[15.4] merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit to create the application environment; and	141
f.	[15.5] delivering the application environment to the compute node;.....	141
g.	[15.6] wherein the modifying comprises: upon receiving a write request from the compute node for a sector X, creating an appropriate persistent mapping for sector X; and.....	141
h.	[15.7] writing sector X on the second storage unit.....	142

13.	Claim 16: The method as recited in claim 15 further comprising: modifying the leaf image in response to the compute node's access to the application environment.....	143
14.	Claim 19: The method as recited in claim 15 wherein the compute node comprises a thin-client workstation.	145
15.	Claim 20: The method as recited in claim 15 wherein the root image comprises an operating system.	145
16.	Claim 21: The method as recited in claim 15 wherein the root image is concurrently accessible to a plurality of compute nodes.	146
17.	Claim 22: The method as recited in claim 15 wherein the first storage unit is remotely located from the compute node.....	146
18.	Claim 23: The method as recited in claim 15 wherein the second storage unit is remotely located from the compute node.	146
19.	Claim 24: The method as recited in claim 15 wherein the second storage unit contains a block modification log for the compute node.	147
20.	Claim 25: The method as recited in claim 15 wherein the first storage unit is contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.	147
21.	Claim 26: The method as recited in claim 15 wherein merging occurs at an operational level between the first and second storage units and file system of the compute node.....	148
22.	Claim 29: The method as recited in claim 15 wherein the first storage unit is read only.	148
23.	Claim 30: The method as recited in claim 15 further comprising: creating an intermediate image on a third storage unit between the root image and the leaf image, wherein the intermediate image comprises an application environment service group.	148

B.	Ground 2: The Combination of Birse and Ballew Renders Claims 2, 11, 12, 18, 27, and 28 Obvious.....	149
1.	Motivations to Combine Birse and Ballew.....	149
2.	Claim 2: The system as recited in claim 1 wherein the compute node comprises a server.	151
3.	Claim 11: The system as recited in claim 1 wherein the system operates in a high performance computing cluster.	151
4.	Claim 12: The system as recited in claim 1 wherein the system operates in a grid computing cluster.	154
5.	Claim 18: The method as recited in claim 15 wherein the compute node comprises a server.	155
6.	Claim 27: The method as recited in claim 15 wherein the method operates in a high performance computing cluster.	156
7.	Claim 28: The method as recited in claim 15 wherein the system operates in a grid computing cluster.....	156
C.	Ground 3: The Combination of Birse and Wu Renders Claims 8 and 24 Obvious.	156
1.	Motivations to Combine Birse and Wu	156
2.	Claim 8: The system as recited in claim 1 wherein the second storage unit contains a block modification log for the compute node.	158
3.	Claim 24: The method as recited in claim 15 wherein the second storage unit contains a block modification log for the compute node.	159
D.	Ground 4: The Combination of Birse and Kleinschnitz Renders Claims 9, 17, and 25 Obvious.	160
1.	Motivations to Combine Birse and Kleinschnitz.....	160
2.	Claim 9: The system as recited in claim 1 wherein the first storage unit in contained within a first partition on a hard disk	

	and the second storage unit is contained within a second partition on the hard disk.	162
3.	Claim 17: The method as recited in claim 15 further comprising: reconciling the root image and the leaf image to form a new root image.	163
4.	Claim 25: The method as recited in claim 15 wherein the first storage unit is contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.	165
XIII.	SECONDARY CONSIDERATIONS	165
XIV.	CONCLUSION.....	166

I, Erez Zadok, Ph.D., declare that:

I. ASSIGNMENT

1. I have been retained by American Airlines, Inc. and Southwest Airlines Co. (collectively “Petitioners”) as an independent expert consultant in this proceeding before the United States Patent and Trademark Office (“PTO”).

2. My consulting company, Zadoks Consulting Services, is being compensated for my time at my standard consulting rate. I am also being reimbursed for expenses that I may incur during the course of this work.

3. My compensation is in no way contingent on the nature of my findings, the presentation of my findings in testimony, or the outcome of this or any other proceeding. I have no other interest in this proceeding.

4. I have been asked to consider whether certain references disclose or suggest the features recited in the claims of U.S. Patent No. 7,721,282 (the “’282 patent”) (EX1001).¹ My opinions are set forth below.

II. BACKGROUND AND QUALIFICATIONS

5. I am a Professor in the Computer Science Department at Stony Brook University (part of the State University of New York (“SUNY”) system). I direct

¹ Where appropriate, I refer to exhibits I understand will be attached to the petition for inter partes review of the ’282 Patent (the “Petition”).

the File-systems and Storage Lab (FSL) at Stony Brook's Computer Science Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security/privacy and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

6. I studied at a professional high school in Israel, focusing on electrical engineering ("EE"), and graduated in 1982. I spent one more year at the high school's college division, receiving a special Certified Technician's degree in EE. I then went on to serve in the Israeli Defense Forces for three years (1983–1986). I received my Bachelor of Science degree in computer science ("CS") in 1991, my Master's degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

7. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting research on operating systems, file and storage systems, distributed and networked

systems, security, and other topics. I also assisted the CS department's computer administrators in managing the department's computers, which included storage, IT, networking, and cyber-security related duties.

8. In 1991, I joined Columbia University's CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well as network devices setup; this included many software and hardware upgrades, device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer (FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

9. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields of security and encryption, efficiency, reliability, and failover. I continued to work

part-time as a systems administrator at the CS department, and eventually I was asked to serve as manager to the entire information technology (“IT”) staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

10. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. “zero-day malware”), using data mining and rule-based detection. This work led to several highly cited papers (over 1,600 citations for one of the papers alone) and two patents. I also became a Teaching Assistant (“TA”) for a first-ever Computer Security course given at Columbia University’s CS department with Dr. Matt Blaze as instructor.

11. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies, and then became the Director of Software Development—overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to

2019, I have consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications' data security in the cloud.

12. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department. My day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late 2019, I've been a member of the department's Executive Committee that also oversees all IT operations.

13. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters. In August 2024, I took over as the

department's Graduate Program Director, overseeing the entire graduate CS program (700-800 students annually on average).

14. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (*e.g.*, NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (*e.g.*, MySQL, PostgreSQL), Web servers (*e.g.*, Apache), and mail servers; user access control (*e.g.*, NIS, LDAP), backups and restores, snapshot policies, and more. I've personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

15. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (*e.g.*, NFS). And in the System Administration

undergraduate course, I covered many topics such as networking, storage, backups, and configuring complex applications such as mail, web, and database servers.

16. My research often investigates computer systems from many angles: security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

17. Since joining Stony Brook University in 2001, my group in the File-systems and Storage Lab (“FSL”) has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write (“COW”) versioning file system, a tracing file system useful to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage,

network, host, etc.). I continue to maintain and release newer versions of some of these file systems and software.

18. I have published over 120 refereed publications (in ACM, IEEE, USENIX, and more). To date, my publications have been cited more than 10,000 times (as per Google Scholar as of March 27, 2025). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled “Linux NFS and Automounter Administration” (Sybex, 2001), covering systems administration topics related to network storage and data security.

19. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called “am-utils” since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software projects for Linux, FreeBSD, and/or Sun Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM’s public release of eCryptfs, now part of Linux. Packet General Networks, for whom I have provided consulting services between 2009 and 2019, licensed another

encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent shadow copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data among several computers, among other features.

20. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering, and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

21. Moreover, in an undergraduate course titled “Advanced Systems Programming in Unix/C,” I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students

were exposed to these principles directly through my teaching and research at Stony Brook University.

22. In another undergraduate course, System Administration, I taught network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection (“CDP”), NAS and SAN, and NFS.

23. Overall, in addition to the aforementioned experience, my technical experience relevant to the '282 Patent at the time of the alleged invention included the configuration, use, research, development, and/or programming of the following technologies and systems: data storage devices, drivers, and services; operating systems layers including (virtual) block layer, file systems, and virtual file systems; copy-on-write storage system and file systems (*e.g.*, Unionfs), including snapshots; commercial and open-source storage appliances (including from Netapp); and caching techniques in one or multi-tier systems.

24. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award, several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY

Chancellor's Excellence in Teaching award, and the 2022 SUNY Chancellor's Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021, I was named an ACM Distinguished Member for "Outstanding Scientific Contributions to Computing."

25. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC'20); serving as the co-chair for USENIX File and Storage Technologies (FAST'15) in 2015 and on the FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage ("TOS") journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM's TOS journal.

26. I am a named inventor on four patents, two titled "Systems and Methods for Detection of New Malicious Executables" (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two more titled "Multi-Tier Caching," (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

27. I have been disclosed as a testifying expert in 19 cases (including *inter partes* review (“IPR”) proceedings) in the past four years. I have been deposed 13 times and testified in trial twice.

28. A copy of my *curriculum vitae* is attached as EX1004.

III. MATERIALS CONSIDERED

29. The opinions contained in this declaration are based on the documents I reviewed, my professional judgment, as well as my education, experience, and knowledge regarding electrical engineering, computer science, video and audio and communications sciences and technologies.

30. In forming my opinions expressed in this declaration, I reviewed the following materials:

Exhibit	Description
1001	U.S. Patent No. 7,721,282 B1, entitled “Block-level I/O subsystem for distributed application environment management,” issued on May 18, 2010 to Kulkarni, <i>et al.</i> (“the ’282 patent”)
1002	File History for U.S. Patent No. 7,721,282
1004	Dr. Erez Zadok Curriculum Vitae
1005	U.S. Patent No. 7,089,300B1, entitled “Method and Apparatus for Administering the Operating System of a Net-Booted Environment,” filed on October 18, 1999, and issued on August 8, 2006 to Birse, <i>et al.</i> (“Birse”)
1006	U.S. Patent Application Publication No. 2005/0235092, entitled “High Performance Computing System and Method,” filed on April 15, 2004, and published on October 20, 2005 to James D. Ballew, <i>et al.</i> (“Ballew”)

Exhibit	Description
1007	U.S. Patent No. 6,981,114, entitled “Snapshot Reconstruction From An Existing Snapshot and One or More Modification Logs,” filed on October 16, 2002, and issued on December 27, 2005 to Weibao Wu, <i>et al.</i> (“Wu”)
1008	U.S. Patent Application Publication No. 2003/0191911, entitled “Using Disassociated Images for Computer and Storage Resource Management,” filed on April 2, 2003, and published on October 9, 2003 to Donald Kleinschnitz, Jr., <i>et al.</i> (“Kleinschnitz”)
1009	File History for U.S. Patent Application No. 11/026,622 (“the 622 Application”, now abandoned)
1011	Microsoft Computer Dictionary, 3rd ed., 1997, excerpts (“Microsoft Computer Dictionary”)
1012	Internet Small Computer Systems Interface (iSCSI), RFC 3720, IETF, April 2004, https://www.rfc-editor.org/rfc/pdf/rfc3720.txt.pdf (“RFC-3720”)
1013	File System Design for an NFS File Server Appliance, Dave Hitz <i>et al.</i> , Network Appliance, USENIX 1994 (“Hitz”)
1014	SnapMirror: File System Based Asynchronous Mirroring for Disaster Recovery, Hugo Patterson <i>et al.</i> , Network Appliance, USENIX 2002 (“Patterson I”)
1015	U.S. Patent 6,668,264B1 to Patterson <i>et al.</i> , filed on Apr. 3, 2001, issued on Dec. 23, 2003, titled “Resynchronization of a target volume with a source volume” (“Patterson II”)
1016	U.S. Patent 7,334,095B1 to Fair <i>et al.</i> filed on Apr. 30, 2004 (“Fair”)
1017	U.S. Patent 8,209,680B1 to Le <i>et al.</i> , Filed June 30, 2003, Issued June 26, 2012, titled “System and method for disk imaging on diverse computers” (“Le”)

31. Exhibits 1011-1017 are true and correct copies of the identified documents that I personally retrieved, copied, own, and/or scanned for purposes of submission with this IPR proceeding. I have also reviewed any other materials I refer to in this declaration in support of my opinions.

32. All of the opinions contained in this declaration are based on the documents I reviewed and my knowledge, experience, and professional judgment. My opinions have also been guided by my appreciation of how a person of ordinary skill in the art would have understood the claims and the specification of the '282 Patent at the time of the alleged invention, which I have been asked to initially consider as no earlier than December 30, 2004 (the filing date of U.S. Patent Application 11/026,622, EX1009). My opinions reflect how one of ordinary skill in the art would have understood the '282 Patent, the prior art to the patent, and the state of the art at the time of the alleged invention.

33. Based on my experience and expertise, it is my opinion that certain references disclose and/or suggest all the features recited in claims 1-30 ("challenged claims") of the '282 Patent, as I discuss in detail below.

IV. LEVEL OF ORDINARY SKILL IN THE ART

34. I have been informed and understand that, in the context of an invalidity analysis, a person having ordinary skill in the art ("POSITA") is not a specific real individual, but a hypothetical person who looks to prior art at the time of the

invention. I further understand that the factors that may be considered in determining the level of ordinary skill include: (1) the level of education and experience of persons working in the field; (2) the types of problems encountered in the field; (3) the teachings of the prior art regarding solutions to such problems; (4) the sophistication of the technology at the time of the alleged invention; and (5) the rapidity of innovation. I understand that these factors need not all be considered for the analysis, and that one or more of these factors may control.

35. I was asked to provide my opinion on the level of one of ordinary skill in the art with respect to the alleged invention of the '282 patent no earlier than December 30, 2004. In my opinion, a POSITA would have had a bachelor's degree in computer science, computer engineering, or an equivalent degree. A POSITA would also have had approximately two years of experience working in the fields of networked data storage, file systems, and operating systems. Additional experience might substitute for less education and *vice versa*. It is also my opinion that these qualifications would have been the same on March 30, 2006 (the filing date of the '282 Patent).

36. As of December 30, 2004, I met, and in fact exceeded, the qualifications of a person of ordinary skill in the art. To be clear, all of my opinions in this declaration are from the perspective of one of ordinary skill in the art as I have defined it here during the relevant timeframe.

V. RELEVANT LEGAL STANDARDS

37. I am not an attorney and offer no legal opinions, but in the course of my work I have had experience studying and analyzing patents and patent claims from the perspective of a person skilled in the art.

38. For the purposes of my analysis and conclusions expressed in this declaration, I have applied the legal principles described in the following paragraphs.

39. Petitioners' counsel has informed me, and I understand, that for the prior art to inherently disclose a claimed limitation, the prior art need not expressly disclose the limitation, so long as the claimed limitation necessarily flows from a disclosure in the prior art. I also understand that it is acceptable to examine evidence outside the prior art reference (extrinsic evidence) in determining whether a feature, while not expressly discussed in the reference, is necessarily present in that reference.

40. Petitioners' counsel has informed me, and I understand, that a patent claim can be considered to have been obvious to a POSITA at the time the application was filed. I am informed that this means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a POSITA at the time of the invention.

41. I have been informed by Petitioners' counsel and understand that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- The level of ordinary skill in the art at the time the application was filed;
- The scope and content of the prior art; and
- What differences, if any, existed between the claimed invention and the prior art.

42. I have been informed by Petitioners' counsel and understand that a single reference can render a patent claim obvious if any differences between that reference and the claims would have been obvious to a POSITA. Alternatively, I understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary skill in the art. In determining whether a combination based on either a single reference or multiple references would have been obvious, it is appropriate to consider, among other factors:

- Whether the teachings of the prior art references disclose known concepts combined in familiar ways, and when combined, would yield predictable results;

- Whether a person having ordinary skill in the art could implement a predictable variation, and would see the benefit of doing so;
- Whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;
- Whether a person having ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;
- Whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- Whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way.

43. I have been informed by Petitioners' counsel and understand that one of ordinary skill in the art has ordinary creativity and is not an automaton. Petitioners' counsel has also informed me that in considering obviousness, obviousness may not be determined using the benefit of hindsight, including hindsight derived from the patent being considered.

44. I further understand that a hypothetical POSITA is assumed to know and to have all relevant prior art in the field of endeavor covered by the patent at issue. I have been informed that there are at least two criteria for determining whether prior art is analogous and thus can be considered prior art: (1) whether the art is from the same field of endeavor, regardless of the problem addressed; and (2) if the reference is not within the field of the patentee's endeavor, whether the reference still is reasonably pertinent to the particular problem with which the patentee is involved. I have also been informed that the field of endeavor of a patent is not limited to the specific point of novelty, the narrowest possible conception of the field, or the particular focus within a given field. I have also been informed that a reference is reasonably pertinent if, even though it may be in a different field from that of the patentee's endeavor, it is one which, because of the matter with which it deals, logically would have commended itself to a patentee's attention in considering their problem.

45. I further understand that, in making a determination as to whether the claimed invention would have been obvious to a POSITA, the U.S. Patent Trial and Appeal Board may consider certain objective factors if they are present, such as: commercial success of products practicing the claimed invention; long-felt but unsolved need; teaching away; unexpected results; copying; and praise by others in the field. These factors are generally referred to as "secondary considerations" or

“objective indicia” of nonobviousness. I understand, however, that for such objective evidence to be relevant to the obviousness of a claim, there must be a causal relationship (called a “nexus”) between the claim and the evidence, and that this nexus must be based on a novel element of the claim rather than something in the prior art. I also understand that even when they are present, secondary considerations may be unable to overcome primary evidence of obviousness (such as motivation to combine with predictable results) that is sufficiently strong.

VI. SUMMARY OF OPINIONS

46. This declaration explains the conclusions I have formed based on my analysis. These conclusions are summarized in the table below:

Ground	Basis	Relied-On References	Claim(s)
1	Obviousness	Birse	1, 3-10, 13-16, 19-26, 29, 30
2	Obviousness	Birse in view of Ballew	2, 11, 12, 18, 27, 28
3	Obviousness	Birse in view of Wu	8, 24
4	Obviousness	Birse in view of Kleinschnitz	9, 17, 25

VII. TECHNOLOGY OVERVIEW

47. In this section, I provide an overview of the background art as understood by a POSITA during the relevant timeframe.

A. General Computer Operations

48. Computers are hardware and software devices that execute instructions. The hardware often includes a central processing unit (“CPU”) or other microprocessors, typically called “processors,” volatile memory (*e.g.*, RAM (random-access memory), or DRAM (dynamic random-access memory)), non-volatile (persistent) storage (*e.g.*, a hard disk or flash drive), and input/output (“I/O”) devices (*e.g.*, network interface card, mouse, keyboard, display). RAM, hard disks, and flash drives are considered, generally, “storage” or “memory” devices because they can store and hold bits of information (whether persistently or not). Computers can have multiple storage devices. Processors are coupled to main memory using a memory bus and coupled to peripherals such as hard disks and network interface cards using one or more I/O busses.

49. Memory and storage devices have considerable differences in their speeds, capacities, and costs. Non-volatile storage devices such as hard disks have the largest capacity, are least expensive per gigabyte, and are slowest (typically operating at millisecond speeds). Volatile DRAM devices are smaller, more expensive, and faster than hard disks (typically operating at microsecond speeds). A CPU’s own memory caches and operations are the smallest, most expensive, and fastest (typically operating at nanosecond speeds). Overall, CPUs are about 1,000× faster than DRAM, which in turn is about 1,000× faster than hard disks.

50. Computer programs are often written in a high-level human-readable language (*e.g.*, C, C++, Java, Perl, PHP, Python), then translated using a compiler or script processor to machine instructions understood by the CPU (*e.g.*, Intel or AMD processor). Machine instructions—generally called “software”—are stored in files on persistent media (*e.g.*, HDD or ROM), loaded into DRAM, and then loaded into the CPU where they can be executed. All modern computers operate in this manner, regardless of their purpose: a small laptop, a server in a data center, a mainframe computer, a personal workstation, a proxy device, a gateway, a firewall, a router, an appliance, a virtual machine, a handheld or mobile device, a smart card, a SD card, etc. Computer systems may run multiple software programs, from background servers and daemons to user applications.

51. Users (including administrators) can login to one or more computers (*e.g.*, desktop, laptop, server) and execute one or more applications (*e.g.*, word processing, web browsing, backup tool). Some computers offer external interfaces and APIs to communicate with certain devices and especially peripherals. For example, both a USB storage device as well as a printer use a well-defined “protocol” to allow users to communicate with them through the main computer.

1. Networking Overview

52. A Local Area Network (“LAN”) is typically a network at a local site where network latencies are short and physical distances between computers are

small. A LAN and its computers are often protected from the rest of the Internet using a firewall. EX1011 (Microsoft Computer Dictionary), 39 (“**LAN** *n.* Acronym for **local area network**. A group of computers and other devices dispersed over a relatively limited area and connected by a communications link that enables any device to interact with any other on the network. LANs commonly include microcomputers and shared resources such as laser printers and large hard disks. The devices on a LAN are known as nodes, and the nodes are connected by cables through which messages are transmitted. *See also* baseband network, broadband network, bus network, collision detection, communications protocol, contention, CSMA/CD, network, ring network, star network, token bus network, token passing, token ring network. *Compare* wide area network.”).

53. Another form of a local area network is one that operates inside a single computer. A “localhost” is “[t]he name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet. *See also* IP address, packet (definition 1), TCP/IP.” EX1011 (Microsoft Computer Dictionary), 40. Often, when users/processes want to communicate with other users/processes, using TCP/IP communications on the same machine, they can use the localhost address: it is considered a trusted, high-speed internal network.

54. Conversely, a Wide-Area Network (“WAN”) refers to computers across the entire Internet, which are often distant from each other, exhibit longer latencies, and afford little protection. EX1011 (Microsoft Computer Dictionary), 66 (“**wide area network** *n.* A communications network that connects geographically separated areas. *Acronym:* WAN.”).

55. A firewall typically intercepts all traffic between computers in a LAN and the WAN, monitors and controls access, and can filter out undesired traffic. Often, LAN computers are far more trusted than WAN computers; therefore, a firewall can be said to separate the trusted network (LAN or localhost) from the untrusted network (WAN). EX1011 (Microsoft Computer Dictionary), 27 (“**firewall** *n.* A security system intended to protect an organization’s network against external threats, such as hackers, coming from another network, such as the Internet. A firewall prevents computers in the organization’s network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the organization’s network, and the proxy server decides whether it is safe to let a particular message or file pass through to the organization’s network.”).

56. A server is a computer or software that provides a service to other computers, called clients. There are many servers on the Internet, for example, Web servers and Mail servers. EX1011 (Microsoft Computer Dictionary), 58 (“**server** *n.*

1. On a local area network (LAN), a computer running administrative software that controls access to the network and its resources, such as printers and disk drives, and provides resources to computers functioning as workstations on the network. 2. On the Internet or other network, a computer or program that responds to commands from a client. For example, a file server may contain an archive of data or program files; when a client submits a request for a file, the server transfers a copy of the file to the client. *See also* client/server architecture. *Compare* client (definition 3).”).

57. A client is typically an end-point computer or software, sometimes used by a user, that communicates across LAN and WAN networks; a client may contact servers to request some service. A user’s desktop, laptop, and handheld device are examples of clients. (Note that a computer can be both a client and a server.) EX1011 (Microsoft Computer Dictionary), 8 (“**client** *n.* 1. In object-oriented programming, a member of a class (group) that uses the services of another class to which it is not related. *See also* inheritance (definition 1). 2. A process, such as a program or task, that requests a service provided by another program—for example, a word processor that calls on a sort routine built into another program. The client process uses the requested service without having to ‘know’ any working details about the other program or the service itself. *Compare* child (definition 1), descendant (definition 2). 3. On a local area network or the Internet, a computer that

accesses shared network resources provided by another computer (called a server). *See also* client/server architecture, server.”).

58. A client/server architecture describes how client computers interact with server computers. EX1011 (Microsoft Computer Dictionary), 8 (“**client/server architecture** *n.* An arrangement used on local area networks that makes use of distributed intelligence to treat both the server and the individual workstations as intelligent, programmable devices, thus exploiting the full computing power of each. This is done by splitting the processing of an application between two distinct components: a ‘front-end’ client and a ‘back-end’ server. The client component is a complete, stand-alone personal computer (not a ‘dumb’ terminal), and it offers the user its full range of power and features for running applications. The server component can be a personal computer, a minicomputer, or a mainframe that provides the traditional strengths offered by minicomputers and mainframes in a time-sharing environment: data management, information sharing between clients, and sophisticated network administration and security features. The client and server machines work together to accomplish the processing of the application being used. Not only does this increase the processing power available over older architectures but it also uses that power more efficiently. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized, multiuser functionality. *See also* distributed intelligence.”).

B. Data Storage Devices and File Systems

59. As mentioned in §VII.A above, computers often include two types of storage or memory devices. The first type is a volatile memory device, called Random Access Memory (“RAM”), that is often used to hold executing programs and their data. EX1011 (Microsoft Computer Dictionary), p. 52 (“RAM”).

60. The second type is a non-volatile, or persistent storage device, such as a hard disk drive (“HDD”). HDDs are used to store programs and data persistently. EX1011 (Microsoft Computer Dictionary), p. 32 (“hard disk”).

61. HDDs are electro-mechanical devices with moving parts. Computers can also have other types of non-volatile storage devices that have no moving parts, also called “solid-state devices,” such as flash memories. EX1011 (Microsoft Computer Dictionary), p. 28 (“flash memory”).

62. Moreover, any single device (*e.g.*, a hard disk) can be logically broken down into several units called “partitions” that otherwise act as their own logical storage area or disk. EX1011 (Microsoft Computer Dictionary), p. 15 (“**disk partition** *n.* A logical compartment on a physical disk drive. A single disk might have two or more logical disk partitions, each of which would be referenced with a different disk drive name. Multiple partitions are divided into a primary (boot) partition and one or more extended partitions.”). A POSITA would have understood that in general, a “partition” is “1. A logically distinct portion of memory or a storage

device that functions as though it were a physically separate unit. 2. In database programming, a subset of a database table or file.” EX1011 (Microsoft Computer Dictionary), p. 47.

63. Above the storage media often lives a logical software layer called a “file system.” File systems provide a convenient abstraction to users and applications of files, folders or directories, and naming. EX1011 (Microsoft Computer Dictionary), p. 26 (“**file system** *n.* In an operating system, the overall structure in which files are named, stored, and organized. A file system consists of files, directories, or folders, and the information needed to locate and access these items. The term can also refer to the portion of an operating system that translates requests for file operations from an application program into low-level, sector-oriented tasks that can be understood by the drivers controlling the disk drives. *See also* driver.”).

1. Files and File Systems

64. A file is a collection of data and attributes, or meta-data, about the file, including the file’s name and extension. EX1011 (Microsoft Computer Dictionary), p. 24 (“**file** *n.* A complete, named collection of information, such as a program, a set of data used by a program, or a user-created document. A file is the basic unit of storage that enables a computer to distinguish one set of information from another. A file is the ‘glue’ that binds a conglomeration of instructions, numbers, words, or

images into a coherent unit that a user can retrieve, change, delete, save, or send to an output device.”).

65. File attributes are also called file meta-data and include security attributes, the name (title), size of a file, timestamps, and more. EX1011 (Microsoft Computer Dictionary), p. 42 (“**meta data** or **metadata** *n.* 1. Data about data. For example, the title, subject, author, and size of a file constitute metadata about the file. *See also* data dictionary, repository.”), p. 24 (“**file attribute** *n.* A restrictive label attached to a file that describes and regulates its use—for example, hidden, system, read-only, archive, and so forth. In MS-DOS, this information is stored as part of the file’s directory entry.”).

66. A file’s name is an important attribute of a file and conveys a lot of useful information about the file’s meaning and purpose. EX1011 (Microsoft Computer Dictionary), p. 25 (“**filename** *n.* The set of letters, numbers, and allowable symbols assigned to a file to distinguish it from all other files in a particular directory on a disk. A file name is the label under which a computer user saves and requests a block of information. Both programs and data have file names and often extensions that further identify the type or purpose of the file. Naming conventions, such as maximum length and allowable characters of a file name, vary from one operating system to another. *See also* directory, path (definition 5).”).

67. A file's path or path-name conveys the location of a file in the name hierarchy and even which file server the file may reside on (*e.g.*, the local C:\ drive or a network drive like N:\). EX1011 (Microsoft Computer Dictionary), p. 48 (“**path** *n.* [...] 5. In file storage, the route followed by the operating system through the directories in finding, sorting, and retrieving files on a disk.”), *id.* (“**pathname** *n.* In a hierarchical filing system, a listing of the directories or folders that lead from the current directory to a file. *Also called:* directory path.”).

68. Moreover, a file's extension conveys vital information about the file's type and hence its meaning and purpose. For example, a file ending with “xls” is known to be a Microsoft Excel spreadsheet, files with “doc” extensions are Microsoft Word files, and those ending with “html” are Web server hyper-text markup language files. EX1011 (Microsoft Computer Dictionary), p. 24 (“**file extension** *n.* See extension (definition 1).”), p. 195 (“**filename extension** *n.* See extension (definition 1).”), p. 186 (“**extension** *n.* 1. A set of characters added to a filename that serves to extend or clarify its meaning or to identify a file as a member of a category. An extension may be assigned by the user or by a program, as, for example, .com or .exe for executable programs that MS-DOS can load and run.”).

69. The (usually non-volatile) storage media that holds the actual bits of file data is often called the “secondary storage” or the “backing storage” (because it is in the “back end”); this is different than “backup storage” that's often used to

backup files for disaster recovery purposes. A “primary storage” is “Random access memory (RAM); the main general-purpose storage region to which the microprocessor has direct access. A computer’s other storage options, such as disks and tape, are called *secondary storage* or (sometimes) *backing storage*.” EX1011 (Microsoft Computer Dictionary), pp. 49-50. A “secondary storage” is “Any data storage medium other than a computer’s random access memory (RAM)-typically tape or disk. *Compare* primary storage.” EX1011 (Microsoft Computer Dictionary), p. 57.

70. Above the storage media often lives a logical software layer called a “file system.” File systems provide a convenient abstraction to users and applications of files, folders or directories, and naming. EX1011 (Microsoft Computer Dictionary), p. 26 (“**file system** *n.* In an operating system, the overall structure in which files are named, stored, and organized. A file system consists of files, directories, and the information needed to locate and access these items. The term can also refer to the portion of an operating system that translates requests for file operations from an application program into low-level, sector-oriented tasks that can be understood by the drivers controlling the disk drives. *See also* driver.”).

71. The “FAT file system” is “[t]he system used by MS-DOS to organize and manage files. The FAT (file allocation table) is a data structure that MS-DOS creates on the disk when the disk is formatted. When MS-DOS stores a file on a

formatted disk, the operating system places information about the stored file in the FAT so that MS-DOS can retrieve the file later when requested. The FAT is the only file system MS-DOS can use; OS/2, Windows NT, and Windows 95 operating systems can use the FAT file system in addition to their own file systems (HPFS, NTFS, and VFAT, respectively). *See also* file allocation table, HPFS, NTFS, OS/2, VFAT, Windows 95, Windows NT.”). EX1011 (Microsoft Computer Dictionary), pp. 22-23.

72. Finally, before any storage media or partition can be used to access files and folders, it must be formatted. To “format” is “2. To prepare a disk for use by organizing its storage space into a collection of data ‘compartments,’ each of which can be located by the operating system so that data can be sorted and retrieved. When a previously used disk is formatted, any preexisting information on it is lost.” EX1011 (Microsoft Computer Dictionary), p. 30 (“Format” definition 2).

C. Networked and Distributed Storage Systems

73. As discussed in §VII.B above, storage devices provide raw storage to computers. One of the most popular storage device protocols is SCSI, which is an “[a]cronym for Small Computer System Interface) a standard high-speed parallel interface defined by the X3T9.2 committee of the American National Standards Institute (ANSI). A SCSI interface is used to connect microcomputers to SCSI peripheral devices, such as many hard disks and printers, and to other computers

and local area networks. *Compare* ESDI, IDE.” EX1011 (Microsoft Computer Dictionary), p. 55.

74. A “SCSI network” is “[a] set of devices on a SCSI bus, which acts like a local area network. *See also* SCSI.” EX1011 (Microsoft Computer Dictionary), p. 56.

75. Indeed, the SCSI protocol was fairly similar to other network protocols, that the IETF began standardizing it in the early 2000s. iSCSI is “a transport protocol for Internet Small Computer Systems Interface (iSCSI) that works on top of TCP. The iSCSI protocol aims to be fully compliant with the standardized SCSI architecture model. ... As system interconnects move from the classical bus structure to a network structure, SCSI has to be mapped to network transport protocols. IP networks now meet the performance requirements of fast system interconnects and as such are good candidates to ‘carry’ SCSI.” EX1012 (RFC 3720), p. 1.

76. File systems can appear as local volumes (*e.g.*, the C:\ drive in Windows), but they can also be networked, where the physical storage resides on a remote file server. Network file systems allow files and volumes to be shared and accessible from multiple networked computers. EX1011 (Microsoft Computer Dictionary), p. 46 (“**Network File System** *n.* *See* NFS.”), p. 45 (“**NFS** *n.* Acronym for **Network File System**. A distributed file system developed by Sun Microsystems,

Inc. that allows users of Windows NT and UNIX workstations to access remote files and directories on a network as if they were local. *Acronym: NFS.*”).

77. Even storage media can be networked into a large cluster of devices, that can export virtual block devices to individual hosts, called a “storage area network” (“SAN”).

78. When the amount of data to access grew large enough, and especially if the data had to be concurrently accessed from multiple users/applications, storage researchers and companies leveraged the principles of distributed systems to improve the performance, resiliency, and fault-tolerance of the storage system.

79. A single computer’s performance is limited by the physical maximum speeds of its network, processor, memory, and I/O devices. A distributed system, however, improves overall performance by using multiple computers concurrently to process many requests, thus increasing the aggregate throughput manyfold.

80. A single computer also represents a single point of failure: if that one computer is down, off, or broken into, then the service it offers is unavailable. A distributed system, conversely, improves resiliency and security by ensuring that the failure of any one (or even a few) components do not affect the overall service provided. This principle is called “fault tolerance.” EX1011 (Microsoft Computer Dictionary), p. 23 (“**fault tolerance** *n.* The ability [] to respond to a catastrophic event or fault, such as a power outage or a hardware failure, in a way that ensures

that no data is lost and any work in progress is not corrupted. [] In a fault-tolerant network, the system has the ability either to continue the system's operation without loss of data or to shut the system down and restart it, recovering all processing that was in progress when the fault occurred.”).

81. NFS is but one example of a networked or distributed file system. Many complex software systems have been designed to operate in a distributed manner, where they are more resilient to failures and scale better than software running on a single node.

82. Generally, a “distributed network” is “[a] network in which processing, storage, and other functions are handled by separate units (nodes) rather than by a single main computer.” EX1011 (Microsoft Computer Dictionary), p. 17.

83. The “Distributed Computing Environment” is “[a] set of standards from the Open Group (formerly the Open Software Foundation) for development of distributed applications that can operate on more than one platform. *Acronym*: DCE. *See also* distributed processing.” EX1011 (Microsoft Computer Dictionary), pp. 16-17.

84. “[D]istributed processing” is “[a] form of information processing in which work is performed by separate computers linked through a communications network. Distributed processing is usually categorized as either plain distributed processing or true distributed processing. Plain distributed processing shares the

workload among computers that can communicate with one another. True distributed processing has separate computers perform different tasks in such a way that their combined work can contribute to a larger goal. The latter type of processing requires a highly structured environment that allows hardware and software to communicate, share resources, and exchange information freely.” EX1011 (Microsoft Computer Dictionary), p. 17.

85. Finally, a “distributed file system” is “[a] file management system in which files may be located on multiple computers connected over a local or wide area network.” EX1011 (Microsoft Computer Dictionary), p. 17.

D. Copy-on-Write and Snapshots

86. Copy-on-Write (CoW) has been used in volatile memory systems inside most operating systems. But CoW can be used elsewhere. Most responsible users would make periodic backups of their data files, often on a daily schedule. Backups are therefore a point-in-time copy of users’ data files. Network Appliance (NetApp), a company producing storage appliances, popularized the idea of using CoW inside storage systems. Netapp called these point-in-time copies “snapshots.” Netapp storage appliances (which I’ve used and maintained for nearly 20 years, *see* §II above) can periodically take read-only snapshots of the storage appliance, thus producing a point-in-time read-only image of the storage system. Snapshots can be taken more frequently (*e.g.*, every hour or even few minutes). Snapshots also save

on disk space, because data blocks/files/sectors that have no changed blocks between snapshots, are shared across multiple snapshots; this is a form of data “deduplication” that can save a lot of space compared to, say, taking full (and even incremental file-based) backups periodically.

87. Netapp’s appliances perform CoW and manage snapshots at the (disk) block layer, but they include a full-fledged file system layer on top of the block layer. That file system can be accessible over the network using NFS (for Unix hosts) or CIFS/Samba (for Windows hosts). See §VII.B.1 above. Netapp’s CoW engine/driver therefore resides between the file system layer and the actual storage devices.

88. Netapp designed a file system optimized for snapshots and network access, using CoW. “WAFL (Write Anywhere File Layout) ... is a file system designed specifically to work in an NFS appliance. The primary focus is on the algorithms and data structures that WAFL uses to implement Snapshots™, which are read-only clones of the active file system. WAFL uses a copy-on-write technique to minimize the disk space that Snapshots consume.” EX1013 (Hitz), p. 4.

89. Snapshots can be created and deleted as needed, offering space saving and improved reliability. “WAFL’s primary distinguishing characteristic is Snapshots, which are read-only copies of the entire file system. WAFL creates and deletes Snapshots automatically at prescheduled times, and it keeps up to 20

Snapshots on-line at once to provide easy access to old versions of files. Snapshots use a copy-on-write technique to avoid duplicating disk blocks that are the same in a Snapshot as in the active file system. Only when blocks in the active file system are modified or removed do Snapshots containing those blocks begin to consume disk space. Users can access Snapshots through NFS to recover files that they have accidentally changed or removed, and system administrators can use Snapshots to create backups safely from a running system. In addition, WAFL uses Snapshots internally so that it can restart quickly even after an unclean system shutdown.” EX1013 (Hitz), p. 6.

90. “The FAServer has commands to let system administrators create and delete Snapshots, but it creates and deletes most Snapshots automatically.” EX1013 (Hitz), p. 7.

91. Netapp maintains meta-data inside files as well as allocation bitmaps. **“3.2. Meta-Data Lives in Files []** Like Episode, WAFL stores meta-data in files. WAFL’s three meta-data files are the inode file, which contains the inodes for the file system, the block-map file, which identifies free blocks, and the inode-map file, which identifies free inodes. The term ‘map’ is used instead of ‘bit map’ because these files use more than one bit for each entry. The block-map file’s format is described in detail below.” EX1013 (Hitz), p. 8.

92. Indeed, files can transparently map to multiple data and meta-data blocks in WAFL. When a file is modified, any of its blocks that need to change are copied first, using standard CoW techniques. “Figure 2 ... shows that files are made up of individual blocks and that large files have additional layers of indirection between the inode and the actual data blocks. In order for WAFL to boot, it must be able to find the root of this tree, so the one exception to WAFL’s write-anywhere rule is that the block containing the root inode must live at a fixed location on disk where WAFL can find it.” EX1013 (Hitz), pp. 9-10, Figure 2 (reproduced below).

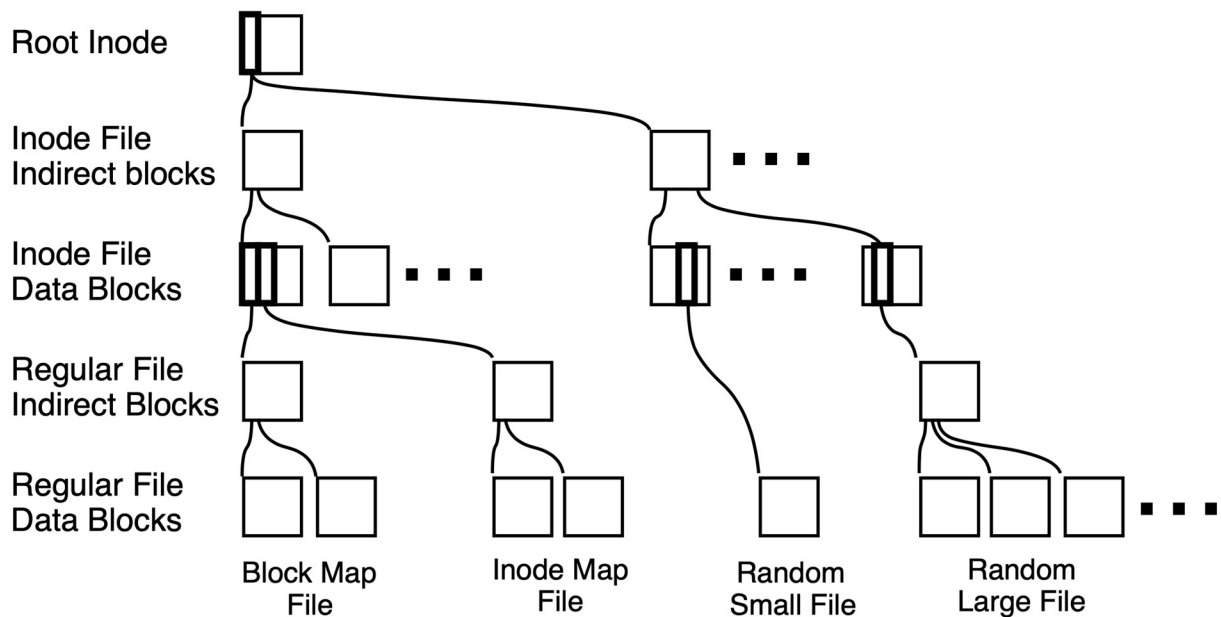


Figure 2
A more detailed view of WAFL's tree of blocks.

93. Netapp details how snapshot works in the presence of file modifications. “**3.4. Snapshots** ... Understanding that the WAFL file system is a tree of blocks rooted by the root inode is the key to understanding Snapshots. To

create a virtual copy of this tree of blocks, WAFL simply duplicates the root inode. Figure 3 shows how this works. Figure 3(a) is a simplified diagram of the file system in Figure 2 ... Figure 3(b) shows how WAFL creates a new Snapshot by making a duplicate copy of the root inode. ... Figure 3(c) shows what happens when a user modifies data block D. WAFL writes the new data to block D' on disk, and changes the active file system to point to the new block. The Snapshot still references the original block D which is unmodified on disk. Over time, as files in the active file system are modified or deleted, the Snapshot references more and more blocks that are no longer used in the active file system. The rate at which files change determines how long Snapshots can be kept on line before they consume an unacceptable amount of disk space.” EX1013 (Hitz), pp. 10-11, Figure 3 (reproduced below).

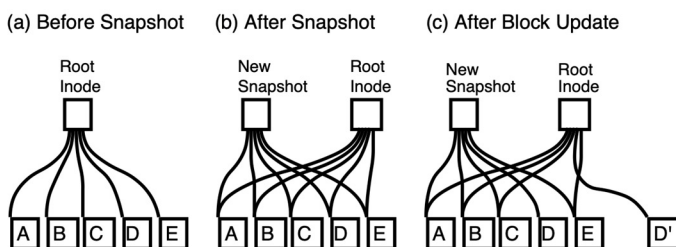


Figure 3

WAFL creates a Snapshot by duplicating the root inode that describes the inode file. WAFL avoids changing blocks in a Snapshot by writing new data to new locations on disk.

94. Initially, Netapp’s snapshots were all read-only, with modifications allowed only at the top of the snapshot stack. This created a linear set of several read-only snapshots followed by the primary writeable storage view, further visible

to applications as a file system. Netapp's next evolution was to permit one to create writeable snapshots, by utilizing CoW of a read-only snapshot.

95. Netapp's developed a "system and method for creating a writable clone of a read-only volume. ... [A] base snapshot is generated on a source volume on a source storage system and is duplicated as a read-only base snapshot replica on a target volume on a destination storage system. A copy ('clone') is then substantially instantaneously created from the read-only base snapshot replica, thereby creating a writable clone of a read-only volume." EX1016 (Fair), 6:18-41.

96. With writeable clones enabled, Netapp filers could now create a (non-linear) tree of snapshots, some read-only and some cloned and writeable. "[M]ultiple clones 1350 may be spawned from the same replica base snapshot 1320; in this case, information is recorded by setting the same snapshot bit in bit fields in both clone's storage label 30 files 1390." EX1016 (Fair), 21:27-31.

97. Now that Netapp had both read-only and writeable snapshots, the next evolution was to transmit a snapshot's data from one networked filer appliance to another, thus creating an identical mirror of the source filer.

98. "Computerized data has become critical to the survival of an enterprise. Companies must have a strategy for recovering their data should a disaster such as a fire destroy the primary data center." EX1014 (Patterson I), p. 2.

99. “Mirrors keep backup data on-line and fully synchronized with the primary store, but they do so at a high cost in performance (write latency) and network bandwidth.” EX1014 (Patterson I), p. 2.

100. “SnapMirror [is] a technology which implements asynchronous mirrors on Network Appliance filers. SnapMirror periodically transfers self-consistent snapshots of the data from a source volume to the destination volume. The mirror is on-line, so disaster recovery can be instantaneous. Users set the update frequency. If the update frequency is high, the mirror will be nearly current with the source and very little data will be lost when disaster strikes. But, by lowering the update frequency, data managers can reduce the performance and network cost of maintaining the mirror at the risk of increased data loss.” EX1014 (Patterson I), pp. 2-3.

101. “During a write episode, WAFL allocates disk space for all the dirty data in the cache and schedules the required disk I/O.” EX1013 (Hitz), p. 12. “A file system with an NVRAM write cache does all the same steps, except that it copies modified data into NVRAM instead of waiting for the data to reach disk.” EX1013 (Hitz), p. 13. “WAFL’s technique for keeping Snapshot data self consistent is to mark all the dirty data in the cache as ‘IN_SNAPSHOT.’” EX1013 (Hitz), p. 17.

102. “WAFL caches inode data in two places: in a special cache of in-core inodes, and in disk buffers belonging to the inode file. When it finishes write

allocating a file, WAFL copies the newly updated inode information from the inode cache into the appropriate inode file disk buffer, and clears the IN_SNAPSHOT bit on the in-core inode.” EX1013 (Hitz), p. 17.

103. “The requested block is then retrieved from disk and stored in a buffer cache of the memory as part of a buffer tree of the file. The buffer tree is an internal representation of blocks for a file stored in the buffer cache and maintained by the file system.” EX1016 (Fair), 3:1-5.

104. “[P]eriodic updates use the primary volume as a giant write cache and it has long been known that write caches can reduce I/O traffic.” EX1014 (Patterson I), p. 8.

105. “[W]e used two Network Appliance F760 filers directly connected via Intel GbE. Each utilized an Alpha 21164 processor running at 600 MHz, with 1024 MB of RAM plus 32 MB non-volatile write cache.” EX1014 (Patterson I), p. 12.

106. Any storage system with snapshots would include at least one complete “base” or “root” and one or more incremental (leaf) images, called “snapshots.” Because snapshots naturally contain incomplete information, they alone cannot be mounted as a file system. Any blocks not included in one snapshot must be located in a previous point in time snapshot; and those not found there have to be located in the next previous snapshot; and so on until one gets to the base/root image which is by definition complete. Thus, the process of traversing several snapshots all the way

down to the root image would result in merging or unifying one or more disparate snapshots with the root/base image, to produce a merged or unified view of the file system that is consistent and cohesive.

107. EX1015 (Patterson II) describes a system for “[r]esynchronization of a target volume with a source volume” in the presence of snapshots. EX1015 (Patterson II), Title. This is useful to mirror two or more volumes in any direction.

108. Patterson II describes “[a]n improved method and apparatus for quickly and efficiently updating the original source volume and original target volumes after the original source volume has become temporarily unavailable. The original target volume is characterized as a source volume while the original source volume is temporarily unavailable. Transfer lists of different data blocks are generated. Data blocks not originally found on a source are copied to the target. Data blocks included on a target that were not found on the source are removed. By focusing upon specific data blocks, this technique avoids the use of filer overhead and other computational resources that would be expended if the entire volume were recopied.” EX1015 (Patterson II), Abstract.

109. Patterson II’s Figure 2 (reproduced below) describes this process.

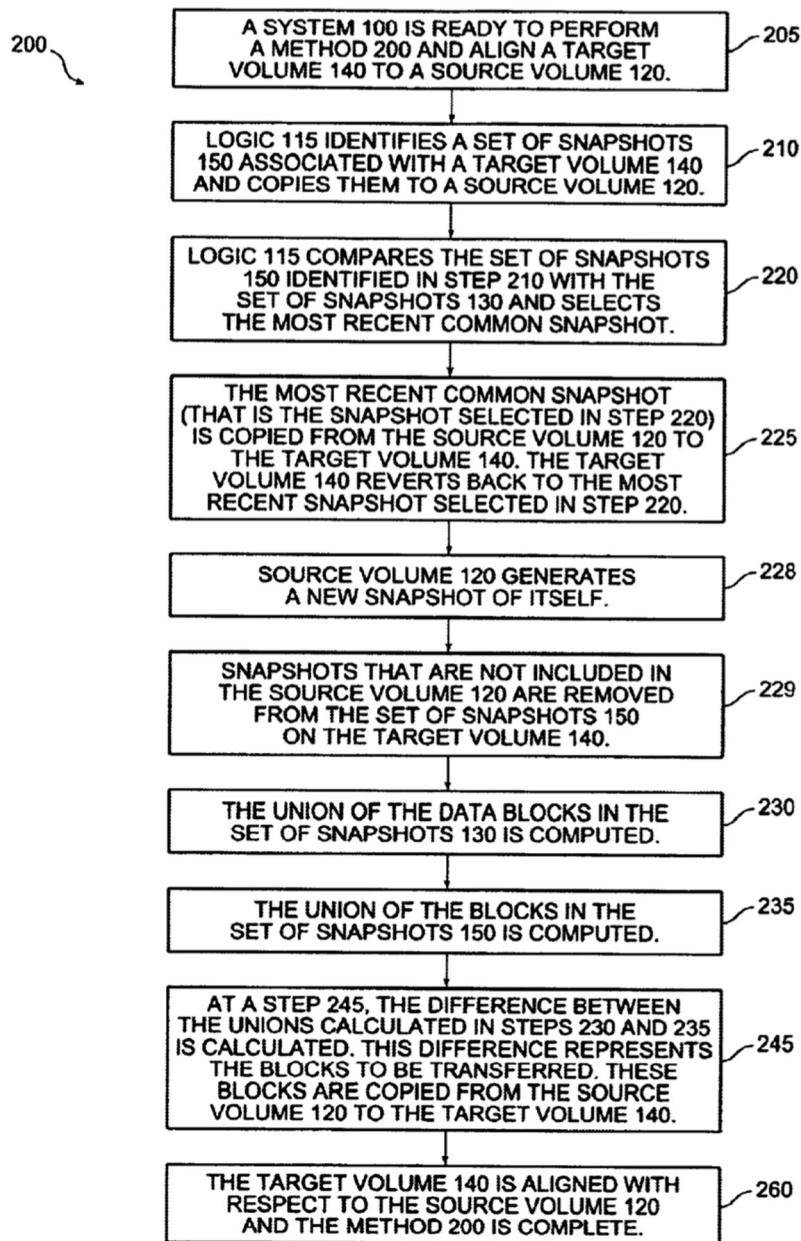


FIG. 2

110. In Figure 2, the system starts by comparing snapshots at the source and destination volumes, to identify a set of differences that have to be transmitted (steps 205-229). The next three steps actually calculate the merged union of the snapshots that need to be transferred, and then transferring those unions.

111. “At a step 230, the union of the data blocks in the set of snapshots 130 is computed. This union will preferably include available and allocated data blocks from any one or more target snapshots 130. This step is preferably performed by the source volume 120. Steps 235 and 245 occur simultaneously with step 230.” EX1015 (Patterson II), 5:6-10.

112. “At a step 235, the union of the blocks in the set of snapshots 150 is computed. This union will preferably include available and allocated data blocks from any one or more target snapshots 150. This step is preferably performed by the source volume 120. Step 235 is performed at the same time as steps 230 and 245.” EX1015 (Patterson II) 5:11-16.

113. “At a step 245, difference between the unions calculated in steps 230 and 235 is calculated. This difference represents the blocks to be transferred. These blocks are copied from the source volume 120 to the target volume 140. In this way, the target volume 140 becomes synchronized with respect to blocks present on the source volume 120.” EX1015 (Patterson II), 5:17-22.

114. Finally, at step 260, the synchronization of the two volumes is complete. A POSITA would easily recognize that the steps Patterson II disclose merge/unify data blocks from multiple snapshots into a set to transmit. This unioning takes place at the block level, because Patterson II’s snapshots are

performed at the block level (the file system resides logically above). EX1015 (Patterson II), Figure 1 (reproduced below).

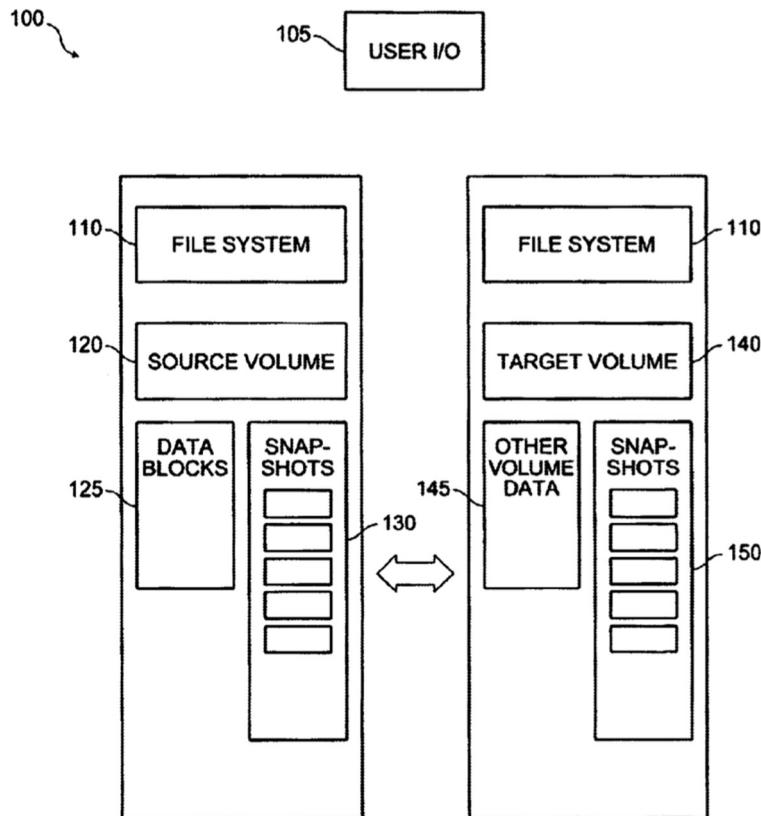


FIG. 1

115. In Patterson II, the software component that performs said unification of snapshots happens to be driven by a file system. “File system software synchronizes the target volume with the source volume. First, the file system software removes snapshots from a target volume if the snapshots are not included in the source volume’s snapshot list. Second, the file system software adds the set of data blocks identified above (that is the set of data blocks that are included in the source volume and not included in the target volume) to its memory. Lastly, the file

system software adds snapshots to the target volume if the snapshots are included in the source volume's snapshot list and not in the target volume's snapshot list. At this point, the target volume includes the data blocks that are present on the source volume." EX1015 (Patterson II), 2:22-34.

116. Lastly, in Patterson II, "sources and volumes can be synchronized dynamically, using a WAFL (Write Anywhere File Layout) system using RAID (Redundant Arrays of Independent Disks) architecture. However, various other types of file systems involving redundant copies of data can also be used." EX1015 (Patterson II), 2:47-53. A POSITA would know that RAID is software that operates at the block level. *See* §VII.B above.

117. EX1017 (Le) includes a lengthy background section detailing that as of 2003 it was known that a "[a] computer disk can be viewed as a linear list of data blocks called sectors," that "a sector-based image format describes the state of a disk at the sector (or 'block') level," and that "a file's content is stored in one or more sectors, called data sectors." EX1017 (Le), 1:16-20. Le thus confirms that sectors and blocks relate to the same subject matter. Le also describes that "a sparse image file ... is a sector-level representation of a complete disk" and that it "does not store the actual contents of unallocated sectors, since their content is known; it needs to store only information about which sectors are unallocated." EX1017 (Le), 3:1-12.

118. Le also discusses that CoW was well-known. Specifically, Le explains that it was known to “employ the copy-on-write technique to allow a virtual machine to modify its virtual disk without actually modifying its virtual disk file,” such that “[w]hen copy-on-write is enabled for a virtual disk, modifications to the file are stored in a separate file, sometimes called a redo log” that “specifies which sector locations in the original disk were written and contains the modified contents for those locations.” EX1017 (Le), 13:55-64. These redo logs “may also be ‘chained’ as a sequence of ‘delta’ disks, each of which records writes to the virtual disk since a most recent preceding checkpoint,” such that “[t]he first such delta disk thus records changes to the initial state of the virtual disk; the second delta disk records writes after the first delta disk is checkpointed; and so on.” EX1017 (Le), 14:4-9.

E. Virtualization

119. Among computer scientists, there is an axiom stating that “you can solve almost any problem by adding another level of indirection.” Indeed, virtualization is an indirection technique that makes one thing seem like another; virtualization often presents a physical-world artifact as one that appears as an illusion or “fake.” For something to be “virtual”, it is “[o]f or pertaining to a device, service, or sensory input that is perceived to be what it is not in actuality, usually as more ‘real’ or concrete than it actually is.” EX1011 (Microsoft Computer Dictionary), p. 61.

120. Indeed, the Microsoft Computer Dictionary includes 35 examples and definitions of virtualization. EX1011 (Microsoft Computer Dictionary), pp. 61-64.

121. In one example, a “virtual machine” is “[s]oftware that mimics the performance of a hardware device, such as a program that allows applications written for an Intel processor to be run on a Motorola chip. *Acronym: VM.*” EX1011 (Microsoft Computer Dictionary), p. 62.

122. In another example, “virtual memory” is “[m]emory that appears to an application to be larger and more uniform than it is. Virtual memory may be partially simulated by secondary storage such as a hard disk. Applications access memory through virtual addresses, are translated (mapped) by special hardware and software onto physical addresses. *Acronym: VM. Also called disk memory. See also paging, segmentation.*” EX1011 (Microsoft Computer Dictionary), p. 62.

123. In yet another example, “virtual reality” is “[a] simulated 3-D environment that a user can experience and manipulate as if it were physical. The user sees the environment on display screens, possibly mounted in a special pair of goggles. Special input devices, such as gloves or suits fitted with motion sensors, detect the user’s actions. *Acronym: VR.*”. EX1011 (Microsoft Computer Dictionary), p. 63.

124. Virtualization had been applied to storage systems, disk devices, and drivers. A “virtual device driver” is “[s]oftware in Windows 95 that manages a

hardware or software system resource. If a resource retains information from one access to the next that affects the way it behaves when accessed (for example, a disk controller with its status information and buffers), a virtual device driver must exist for it.” EX1011 (Microsoft Computer Dictionary), p. 62.

125. A “virtual disk” or “RAM disk” is “[s]hort for random access memory disk. A simulated disk drive whose data is actually stored in RAM memory. A special program allows the operating system to read from and write to the simulated device as if it were a disk drive. RAM disks are extremely fast, but they require that system memory be given up for their use. Also, RAM disks usually use volatile memory, so the data stored on them disappears when power is turned off. Many portables offer RAM disks that use battery-backed CMOS RAM to avoid this problem. *See also* CMOS RAM. *Compare* disk cache.” EX1011 (Microsoft Computer Dictionary), pp. 52-53, 62.

126. Virtualization is sometimes said to be “logical” representation of physical reality. “Logical” is “[o]f or pertaining to a conceptual piece of equipment or frame of reference, regardless of how it may be realized physically. *Compare* physical.” EX1011 (Microsoft Computer Dictionary), p. 41 (“logical” definition 2).

127. A “logical drive” or “logical device” is “[a] device named by the logic of a software system, regardless of its physical relationship to the system. For example, a single floppy disk drive can simultaneously be, to the MS-DOS operating

system, both logical drive A and drive B.” EX1011 (Microsoft Computer Dictionary), p. 41.

128. Moreover, one can virtualize files and file systems. This was the subject of much of my graduate-level research and became part of my PhD Dissertation. *See* §II above and EX1004 (Zadok Curriculum Vitae).

129. Virtualization, of course, has been applied to networking as well. A “virtual network” is “[a] part of a network that appears to a user to be a network of its own. For example, an Internet service provider can set up multiple domains on a single HTTP server so that each one can be addressed with its company’s registered domain name. *See also* domain name, HTTP server (definition 1), ISP.” EX1011 (Microsoft Computer Dictionary), p. 63.

130. A “virtual LAN” is “[s]hort for **virtual local area network**. A local area network consisting of groups of hosts that are on physically different segments but that communicate as though they were on the same wire. *See also* LAN.” EX1011 (Microsoft Computer Dictionary), p. 62.

131. One of the most famous uses of a (secure) virtual network is a private one. A “virtual private network” (“VPN”) is “[a] set of nodes on a public network such as the Internet that communicate among themselves using encryption technology so that their messages are as safe from being intercepted and understood by unauthorized users as if the nodes were connected by private lines. 2. A wide

area network formed of permanent virtual circuits (PVCs) on another network, especially a network using technologies such as ATM or frame relay. *Acronym:* VPN. *See also* ATM (definition 1), frame relay, PVC.” EX1011 (Microsoft Computer Dictionary), p. 63. A POSITA would have understood definition 2 of VPN to be a form of an overlay network because it is a collection of individual virtual network segments.

VIII. THE '282 PATENT

A. Overview

132. The '282 Patent describes “a system for distributing an application environment to a compute node.” EX1001 ('282 Patent), Abstract. “The system includes a first storage unit for storing blocks of a root image of the compute node and a second storage unit for storing a leaf image comprising new data blocks and changes to the blocks of the root image.” *Id.* A POSITA would have recognized that these leaf images correspond to storage snapshots as I describe in §VII.D above.

133. The '282 Patent then describes that the system “further includes a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node,” where “[t]he union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the leaf image stored on the second storage unit.” *Id.*

134. The '282 Patent describes a background where, “as the internet has expanded and computers have multiplied, the need for clustered computing such as High Performance Computing (HPC) has increased[,]” and that “[c]lustered computing involves multiple compute nodes, usually a server grid, that work together to achieve a common task.” EX1001 ('282 Patent), 1:21-38.

135. The '282 Patent describes the trend towards global file system management: “[i]n large-scale systems such as this, a trend in software deployment is to centralize data management on a globally accessible file system with stateless computing nodes.” *Id.*

136. That led to the need to manage individual nodes' boot software and/or boot images. “A common example of this is Operating System (OS) software image management, where the compute nodes are activated with the distributed application environment by either diskless booting protocols or remote software installation to local storage. Under this architecture, a boot image is required for each compute node in the cluster.” *Id.*

137. OS boot images contain the OS kernel, libraries, applications, and more. “The boot image necessarily contains the kernel; it may additionally contain the application software that is intended to be run on the compute node.” *Id.*

138. The '282 Patent explains a concern with managing so many boot images, many of which may be (nearly or) identical. “The primary concern in

clustered computing is low cluster bring-up time. The software that provides the boot images for the cluster typically stores a master boot image. It may then either pre-create clones of this master image for each such server, or it may create them ‘on the fly.’” EX1001 (’282 Patent), 1:39-57.

139. Thus, “[c]reating a boot image on the fly involves copying the entire contents of the master image, which are typically in the range of 5-15 GB.” *Id.* This can be slow. “Even with a significant amount of bandwidth by today’s standards, this method will result in a large bring-up time.” *Id.*

140. One proposed solution is that “[p]re-creating a boot image for each server is advantageous from the point of view of cluster bring-up time.” *Id.* But such a solution could waste disk/storage space “since one often does not know in advance how many servers will ever be booted, this scheme may result in wasted disk space.” *Id.* Thus, “both [methods] suffer from the same major problem—updating the boot image(s) for the cluster is cumbersome, as it means updating a number of copies of the boot image.” *Id.*

141. The ’282 Patent’s Background section then discusses how “[i]n a branching store file system, a read-only base image (or ‘root’ image) of the application environment is created.” EX1001 (’282 Patent), 1:62-2:5.

142. “The root image is accessible by all compute nodes in the cluster” and “[c]hanges made by a compute node to the root image are stored in a ‘leaf’ image unique to that compute node” (*i.e.*, a snapshot). *Id.*

143. To make this work, a software driver, component, or filter “operates between the compute nodes and the file system(s), which merges the changes recorded on the leaf images with the root image and delivers the result to the appropriate compute node.” *Id.*

144. The benefit is that, “[f]rom the point of view of the compute node, it is running its own unique and cohesive instance of the application environment.” *Id.*

145. Nevertheless, “[w]hile this system allows for creation of boot images on the fly without severely diminishing bring-up time, a separate version of the system must be created for each unique operating system.” *Id.*

146. Therefore, the ’282 Patent argues that “migrating a computing cluster from one operating system to another is much more complicated than simply installing a new root image containing the new OS.” *Id.*

147. After discussing and admitting to this known prior art, the ’282 Patent describes an example system 100 “for distributing an application environment. In a preferred embodiment, system 100 is implemented in a multi-computer system, such as an HPC cluster. In one embodiment, the application environment includes an

operating system. In other embodiments, the application environment may contain other applications.” EX1001 (’282 Patent), 3:62-4:10, Fig. 1 (reproduced below).

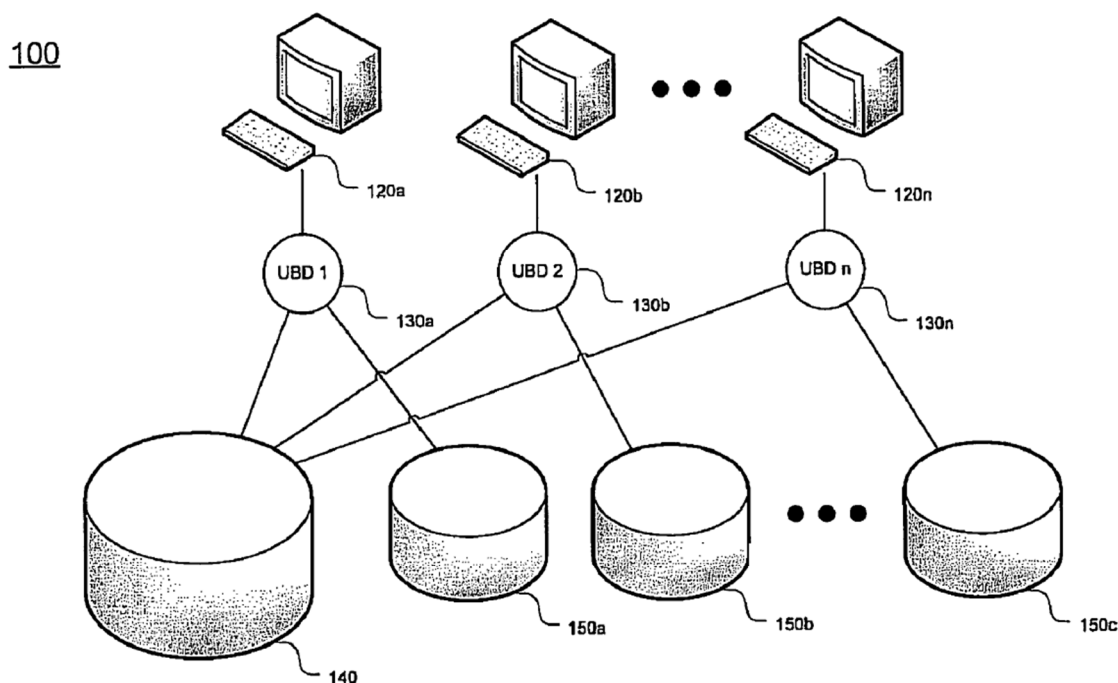


Figure 1

’282 Patent, Figure 1

148. It then describes how the nodes and the storage they use. “System 100 has a number of compute nodes 120a-n coupled to first storage unit [140] and a corresponding second storage unit 150a-n [sic] though a corresponding union block device (UBD) 130a-n. To the compute nodes 120a-n, it appears that they have access to their own version of a distributed application environment. However, a separate and complete boot image is not created and stored for each compute node

120a-n.” *Id.* Of note is the disclosed “UBD”, which I describe in more detail in §VIII.B below.

149. Instead of having separate boot images for each node, “a first storage unit 140” “store[s] blocks of a root image of an application environment. The root image contains data initially common to the compute nodes 120a-n” and “is not changed by the compute nodes 120a-n.” *Id.*, 4:11-16.

150. In one embodiment, compute nodes 120a-n have read-only access to the first storage unit 140. Each compute node 120a-n also has a corresponding second storage unit 150a-n for storing a leaf image, which in the challenged claims includes “new data blocks and changes to the blocks of the root image ” or similar language. *Id.*, 4:14-18, 7:21-24, 8:12-15.

151. In one embodiment, “each compute node 120a-n has a corresponding second storage unit 150a-n for storing a leaf image. The first storage unit 140 and second storage units 150a-n may each be contained on separate physical storage devices, on separate logical spaces on the same storage device, or any combination thereof.” EX1001 (’282 Patent), 4:17-40.

152. “The leaf image may contain blocks of new data, blocks of changed data, or other blocks of data unique to the individual compute node.” *Id.* Thus, “a leaf image will describe the changes made by the respective compute node 120a-n to its instance of the application environment.” *Id.*

153. The '282 Patent states that “[w]ith respect to changes to the root image, only the specific blocks that are changed are stored in the leaf image. ... One compute node (e.g., compute node 120a) desires to make a change to this file which involves a modification of only a few specific blocks of the file (e.g., blocks 4-9). In this example, only the modified blocks (e.g., blocks 4-9) will be stored in the compute node’s leaf image (e.g., leaf image stored on second storage device 150a) plus some small overhead.” EX1001 ('282 Patent), 4:17-40. As shown in FIG. 2, below, the leaf images are also referred to as leaf nodes, which “may be for blocks from configuration files.” EX1001 ('282 Patent), 5:48-52.

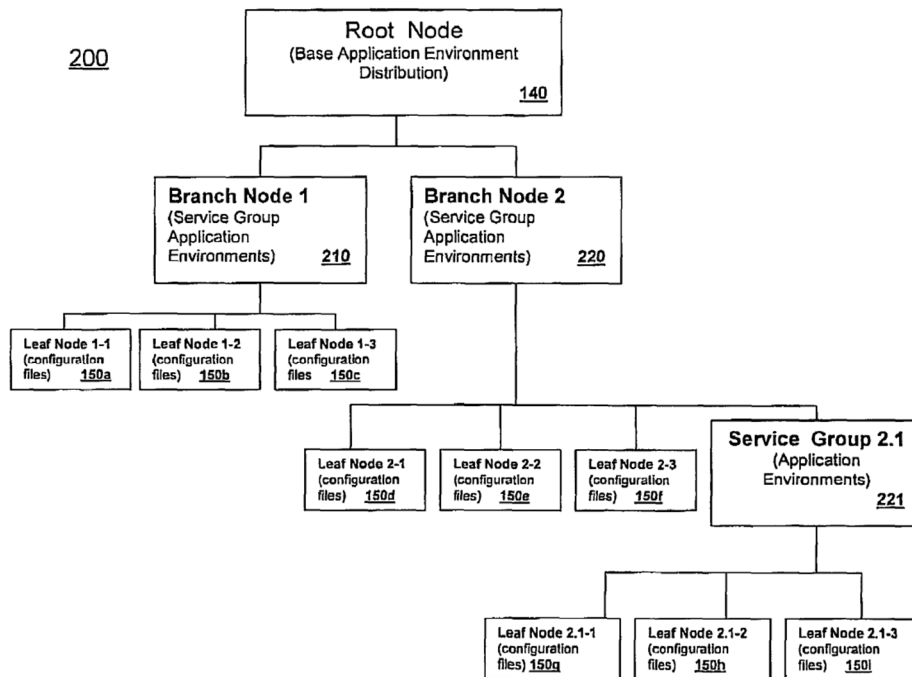


Figure 2

154. The first storage unit 140 and second storage units 150*a-n* may each be contained on separate physical storage devices, on separate logical spaces on the same storage device, or any combination thereof. *Id.*

155. A POSITA would have understood this to claim snapshots as I describe in §VII.D above.

156. Next, the '282 Patent describes a “union block device” (UBD) that logically merges blocks between a root image and one or more leaf images, just as snapshotting systems do. I describe this UBD and its relationship to commonly known snapshots in detail in §VIII.B below. EX1001 ('282 Patent), 4:41-5:10.

B. The '282 Patent's Union Block Device (UBD)

157. I note that the '282 Patent does not define what a UBD is. The patent describes that UBDs may be low-level drivers. “In one embodiment, UBDs 130*a-n* 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 120*a-n*. ... Because UBDs 130*a-n* operate below the file system, they are concerned merely with the blocks of data themselves, rather than files they form.” EX1001 ('282 Patent), 4:42-51. Many storage drivers reside below file systems. *See* §VII.B above.

158. UBDs are said to “determine what leaf image (from the appropriate second storage unit 150) is needed for portions of the application environment that their respective compute nodes 120*a-n* have changed. UBDs 130*a-n* also locate the

portions of the application environment that are not changed by their respective compute nodes 120*a-n*. These portions may reside in the root image. There may also be intermediate images (not depicted in FIG. 1) comprising versions of the root image from which a compute node's instance of the application environment is derived." EX1001 ('282 Patent), 4:52-67. A POSITA would have understood that this is no more than common traversal of snapshots. *See* §VII.D above.

159. Moreover, "UBDs 130*a-n* may also modify the leaf image in response to their respective compute node's access to its instance of the application environment. For example, upon receiving a write request from their respective compute nodes for a sector X, the UBDs 130*a-n* will create an appropriate persistent mapping for sector X and then write sector X onto their respective second storage units 150*a-n*, where sector X can then be modified." EX1001 ('282 Patent), 5:1-11. Again, a POSITA would have understood that the UBD functionality disclosed here is nothing more than typical copy-on-write (CoW). *See* §VII.D above.

160. Lastly, UBDs logically merge different root and leaf images, which are nothing more than snapshots. Thus "Step 330 is merging the blocks of the root image with the blocks of the leaf image to create the application environment. Once the application environment has been created, it will appear to the compute node as one cohesive image rather than a base image plus its additions, deletions, and modifications. To the compute node, it appears that it has access to its own unique

version of an application. However, a separate and complete boot image is not actually stored for the compute node.” EX1001 (’282 Patent), 6:8-16. A POSITA would have recognized this merging functionality to be no different than what prior art snapshot systems already disclosed. *See* §VII.D above.

C. Prosecution History

1. The ’816 Application

161. U.S. Application No. 11/395,816 (the “’816 Application”), which matured into the ’282 Patent, was a continuation-in-part of U.S. Patent Application No. 11/026,622 (the “’622 Application”).

162. The ’816 Application originally contained two independent claims (claims 1 and 17) and 33 claims in total. The two independent claims both recited storing blocks of a root image on a first storage unit and storing a leaf image comprising new data blocks and changes to the blocks of the root image on a second storage unit, and merging the blocks of the root image with blocks of the leaf image [via a union block device in claim 1] to create an application environment. EX1002 (the ’282 Patent File History), 22-26.

163. The PTO issued a First Office Action on July 8, 2009. The Examiner rejected claims 1-33 as obvious over U.S. Patent No. 6,502,238 (“Pavan”) in view of U.S. Patent No. 7,475,274 (“Davidson”). EX1002 (the ’282 Patent File History), 60-82. Pavan was only for teaching a first storage unit for storing blocks of a root

image, a second storage unit for storing blocks of a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, and the Examiner found it would have been obvious to a POSITA at the time of the alleged invention to have combined the “root image” and “leaf image” teachings of Pavan with a “union block device” or management node 105 used for management of HPC clusters teachings of Davidson for “achiev[ing] processing efficiency and reduc[ing] compilation time.” *Id.*

164. The Applicants submitted an Amendment and Response on October 8, 2009. Therein, the two independent claims (claims 1 and 17) were amended to recite “wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block,” to recite “wherein the second storage unit comprises a second non-volatile memory,” and to recite “wherein the union block device comprises a driver.” EX1002 (the ’282 Patent File History), 90-92. For example, claim 1 was amended as follows:

1. (Currently amended) A system for distributing an application environment ~~to a compute node comprising:~~

a compute node comprising a computer system:

a first storage unit for storing blocks of a root image of the compute node, wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;

a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, wherein the second storage unit comprises a second non-volatile memory; and

a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node, wherein the union block device comprises a driver, wherein the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit.

165. In the accompanying remarks, the Applicants stated that the addition of these new “wherein” clauses overcame the Examiner’s July 8, 2009, obviousness rejections, and that Pavan’s “program blocks” of a “block-based programming model” were not blocks of a root image that comprise a computer program as claimed. EX1002 (the ’282 Patent File History), 98-101. The Applicants also argued that Davidson’s HPC management node 102 did not merge blocks of images. *Id.*, 101-103. The Applicants did not expressly raise any of the dependent claim limitations as a grounds for allowance. *Id.*

166. A Notice of Allowance was issued on January 26, 2010. EX1002 (the '282 Patent File History), 108-115. The Notice of Allowance included an Examiner's Amendment agreed to by the Applicants that canceled Claims 10, 11, and 19 and amended Claim 1 to recite "the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node; and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X," and amended Claim 17 to recite "wherein the modifying comprises: upon receiving a write request from the compute node for a sector X, creating an appropriate persistent mapping for sector X; and writing sector X on the second storage unit." *Id.*, 112-114. The Notice of Allowance included a general statement that the prior art of record did not teach these elements of the claims.

167. Thus, the sole purported point of novelty for the claims of the '282 Patent expressly raised during prosecution is the limitation, in each of the independent claims, that the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node; and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X." A POSITA would have understood this additional language to claim snapshots as I describe in §VII.D above.

2. The '622 Application

168. A different Examiner was assigned to the “parent” '622 Application. That Examiner rejected all of the claims initially proffered by the Applicants in an April 13, 2009, first Office Action—as being anticipated by Chang. The Examiner thereafter repeatedly rejected all subsequently pending claims (in Office Actions dated August 24, 2009; March 15, 2010; August 13, 2010; and December 6, 2010) as obvious over Chang in view of U.S. Patent No. 6,751,658 (Haun). EX1009 ('622 Application File History), 50-63, 92-109, 144-167, 203-25, 259-85. Indeed, the Applicants ultimately abandoned the '622 Application with no claims ever being deemed allowable. *Id.*, 294-95.

169. The Examiner assigned to the '622 Application repeatedly rejected every pending claim despite the prosecuting attorneys (who were the same attorneys prosecuting the '816 Application, at least for a significant portion of the time) proffering claims that included several similar elements such as base images and leaf images, as shown below:

1. A branching store file system, comprising:

a root file system comprising a base image of a distributed application, wherein said base image is read-only accessible by compute nodes;

leaf images comprising modified portions of said distributed application for respective ones of said compute nodes, wherein an instance of said distributed application for a first of said compute nodes is based on said base image and at least one of said leaf images; and

a filter that controls access of instances of said distributed application for said compute nodes.

12. (Currently amended) A branching store file system, comprising:

a root file system having a root metadata structure and a root data structure, said entries in said root metadata structure pointing to associated unmodified data in said root data structure; and

a branch modification log for respective ones of compute nodes, comprising:

a branch data structure comprising modified data reflecting changes made to an instance of said root data structure by said branch modification log's respective compute node; and

a branch metadata structure comprising unmodified and modified entries, said unmodified entries pointing to associated portions of said unmodified data in said root data structure, and said modified entries pointing to associated portions of said modified data in said branch data structure, wherein an instance of a distributed application for a first of said compute nodes is instantiated by accessing defined by said branch metadata structure, wherein said root data structure and said branch data structure are on separate nodes.

EX1009 (the '622 Application File History), 25-28, 73-80.

3. Priority Date

170. The '282 Patent claims priority to the '622 Application (filed on December 30, 2004). EX1002 ('282 Patent File History), EX1009 ('622 Application File History). I understand that the Petitioners assert the '282 Patent is not entitled to the priority claim, as the original '622 Application does not ever mention “merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit to create the application environment” nor “upon receiving a write request from the compute node for a sector X, creating an appropriate persistent mapping for sector X” – which are expressly required by all claims of the '282 Patent. EX1001 ('282 Patent), 5:1-10, 7:11-8:65; *compare* EX1001 ('282 Patent) 5:61-6:38, Fig. 3 *with* EX1009 ('622 Application File History), 4-40. Additionally, claims 1-14 of the '282 Patent recite using a “union block device,” which is also not found in the '622 Application. *See* EX1009 ('622 Application File History), 4-40. As stated in §III above, my opinions have been guided by my appreciation of how a POSITA would have understood the claims and the specification of the '282 Patent at the time of the alleged invention, which I have been asked to initially consider as no earlier than December 30, 2004 (the filing date of U.S. Patent Application 11/026,622, EX1009).

IX. THE CLAIMS

171. The tables below include identifiers for the '282 Patent claims and limitations, which are referenced throughout this Declaration.

Reference	Claim Limitation
Claim 1	
[1.0]	1. A system for distributing an application environment comprising:
[1.1]	a compute node comprising a computer system; a first storage unit for storing blocks of a root image of the compute node,
[1.2]	wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;
[1.3]	a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, wherein the second storage unit comprises a second non-volatile memory; and
[1.4]	a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node, wherein the union block device comprises a driver,
[1.5]	wherein the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit;
[1.6]	the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node;

Reference	Claim Limitation
Claim 1	
[1.7]	and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X.
Claim 2	
[2]	2. The system as recited in claim 1 wherein the compute node comprises a server.
Claim 3	
[3]	3. The system as recited in claim 1 wherein the compute node comprises a thin-client workstation.
Claim 4	
[4]	4. The system as recited in claim 1 wherein the root image comprises an operating system.
Claim 5	
[5]	5. The system as recited in claim 1 wherein the root image is concurrently accessible to a plurality of compute nodes.
Claim 6	
[6]	6. The system as recited in claim 1 wherein the first storage unit is remotely located from the compute node.

Claim 7	
[7]	7. The system as recited in claim 1 wherein the second storage unit is remotely located from the compute node.

Claim 8	
[8]	8. The system as recited in claim 1 wherein the second storage unit contains a block modification log for the compute node.

Claim 9	
[9]	9. The system as recited in claim 1 wherein the first storage unit is contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.

Claim 10	
[10]	10. The system as recited in claim 1 wherein the union block device writes sector X on the second storage unit.

Claim 11	
[11]	11. The system as recited in claim 1 wherein the system operates in a high performance computing cluster.

Claim 12	
[12]	12. The system as recited in claim 1 wherein the system operates in a grid computing cluster.

Claim 13	
[13]	13. The system as recited in claim 1 wherein the first storage unit is read only.

Claim 14	
[14]	14. The system as recited in claim 1 further comprising an intermediate image between the root image and the leaf image, wherein the intermediate image comprises an application environment service group.

Claim 15	
[15.0]	15. A method for distributing an application environment comprising:
[15.1]	storing blocks of a root image of a compute node on a first storage unit, wherein the compute node comprises a computer system,
[15.2]	and wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;
[15.3]	storing a leaf image comprising new data blocks and changes to the blocks of the root image on a second storage unit, wherein the second storage unit comprises a second non-volatile memory;
[15.4]	merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit to create the application environment; and
[15.5]	delivering the application environment to the compute node;
[15.6]	wherein the modifying comprises: upon receiving a write request from the compute node for a sector X, creating an appropriate persistent mapping for sector X; and
[15.7]	writing sector X on the second storage unit.

Claim 16	
[16]	16. The method as recited in claim 15 further comprising:

Claim 16	
	modifying the leaf image in response to the compute node's access to the application environment.

Claim 17	
[17]	17. The method as recited in claim 15 further comprising: reconciling the root image and the leaf image to form a new root image.

Claim 18	
[18]	18. The method as recited in claim 15 wherein the compute node comprises a server.

Claim 19	
[19]	19. The method as recited in claim 15 wherein the compute node comprises a thin-client workstation.

Claim 20	
[20]	20. The method as recited in claim 15 wherein the root image comprises an operating system.

Claim 21	
[21]	21. The method as recited in claim 15 wherein the root image is concurrently accessible to a plurality of compute nodes.

Claim 22	
[22]	22. The method as recited in claim 15 wherein the first storage unit is remotely located from the compute node.

Claim 23	
[23]	23. The method as recited in claim 15 wherein the second storage unit is remotely located from the compute node.
Claim 24	
[24]	24. The method as recited in claim 15 wherein the second storage unit contains a block modification log for the compute node.
Claim 25	
[25]	25. The method as recited in claim 15 wherein the first storage unit is contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.
Claim 26	
[26]	26. The method as recited in claim 15 wherein merging occurs at an operational level between the first and second storage units and file system of the compute node.
Claim 27	
[27]	27. The method as recited in claim 15 wherein the method operates in a high performance computing cluster.
Claim 28	
[28]	28. The method as recited in claim 15 wherein the system operates in a grid computing cluster.

Claim 29	
[29]	29. The method as recited in claim 15 wherein the first storage unit is read only.

Claim 30	
[30]	30. The method as recited in claim 15 further comprising: creating an intermediate image on a third storage unit between the root image and the leaf image, wherein the intermediate image comprises an application environment service group.

X. OVERVIEW OF THE PRIMARY PRIOR ART REFERENCES

172. I first note that the prior art references analyzed in the grounds below are analogous to the '282 Patent because they are within the same field of endeavor and/or reasonably pertinent to one or more problems addressed by the '282 Patent. For example, all are related to storage systems, snapshot management, and using copy-on-write techniques.

A. Birse

173. Birse describes a system “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.” EX1005 (Birse), Abstract. Birse describes using “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. *Id.*, 14:56-61.

174. In Birse, “[a]n ‘image’ refers to an electronic representation of an individual disk or a portion thereof.” *Id.*, 4:1-2.

175. In Birse, a “‘thick client’ generally refers to a computer system, such as a personal computer (PC), that includes a hard drive or other local storage medium. Traditionally, such computer systems boot and execute application programs from the local storage medium.” *Id.*, 4:22-26. Conversely, a “‘thin client’ generally refers to a disk-less computer system that relies upon external means, such as a server, to supply it with an operating system and application programs.” *Id.*, 4:27-30.

176. In Birse, “the terms ‘network computer client’ or ‘NC client’ generally refer to a client (thick or thin) that boots by accessing a copy of the operating system over a network. As such, an NC client is not required to be disk-less.” *Id.*, 4:31-35.

177. Moreover, a “‘network computer server’ or ‘NC server’ generally refer to a computer system that services requests of NC clients. For example, an NC server may provide access to a copy of the operating system to an NC client in response to a boot request.” *Id.*, 4:36-40.

178. Then, a “‘network computer system’ or ‘NC system’ refers to a network including one or more NC clients and one or more NC servers.” *Id.*, 4:41-43.

179. Birse’s “‘volume’ generally refers to a fixed amount of storage on a storage medium, such as a disk or tape. In some instances, however, the term may be used as a synonym for the storage medium itself. However, it is possible for a single storage medium to contain more than one volume or for a volume to span more than one storage medium.” *Id.*, 4:44-50.

180. “According to one embodiment ... a network computer (‘NC’) system maintains a copy of the OS that cannot be corrupted by ordinary users of the NC system.” *Id.*, 2:56-59.

181. “Additionally, the NC system may preserve user customizations (e.g., preferences) by maintaining individual, user, storage areas.” *Id.*, 2:59-61.

182. “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.*, 2:62-65.

183. Thus, because “the user’s desktop preferences and other customized settings are 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.” *Id.*, 2:65-3:3.

184. Birse’s “FIG. 1 is a simplified block diagram that conceptually illustrates an NC system 100 according to one embodiment of the present invention” where “the NC system 100 includes an NC client 150 coupled to an NC server 170 via a data communication link 140, such as ... Ethernet link.” *Id.*, 4:65-5:8, FIG. 1 (reproduced below).

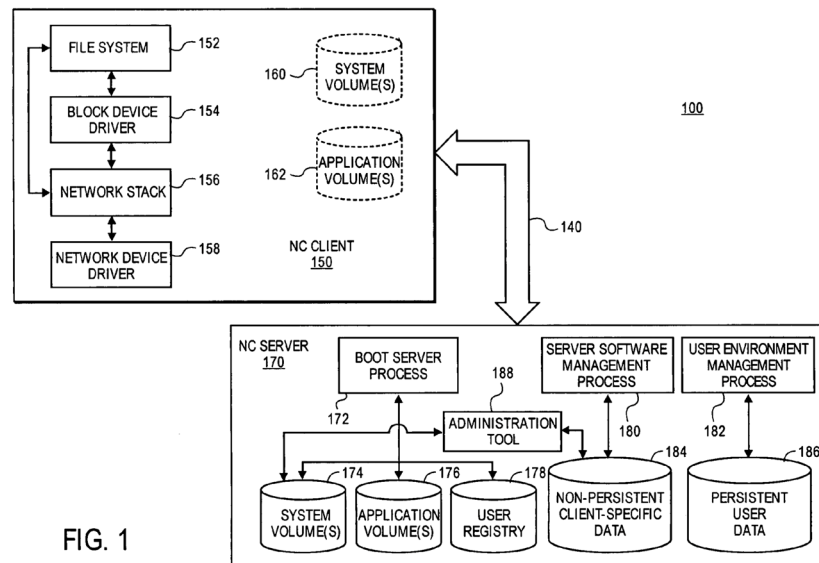


FIG. 1

Birse, Figure 1

185. “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.” *Id.*, 5:9-29.

186. Moreover, “[t]he system volumes 174 include a protected, read-only, master copy of the operating system software. The application volumes 176 include copies of various application programs for use by the NC client 150” and a “user registry 178 [that] is a database of authorized users, user passwords, NC client

hardware addresses, such as Ethernet Media Access Control (MAC) addresses, and NC client network addresses, such as Internet Protocol (IP) addresses.” *Id.*

187. Then, “[t]he non-persistent client-specific data 184 represents temporary data storage that is typically preserved only for the duration of a user session on a particular NC client. The persistent user data 186 represents long-term data storage for user information that is desirable to retain between user sessions, such as preferences, browser bookmarks, and other desktop environment customizations.” *Id.*, 5:9-29.

188. In Birse, “the boot server process 172 manages access to and from the system volumes 174 and the application volumes 176. Additionally, the boot server process 172 performs server-side bootstrapping processing” for example, using the “conventional Bootstrap Protocol (Bootp) server processing that waits for NC clients to appear on the network in accordance with Request For Comments (RFC) 951, ... RFC 2132, and uses the standard extensions format.” *Id.*, 5:46-56.

189. “[T]o facilitate remote NC system administration, such as the installation of new system or application software, the boot server process 172 may serve application and system images that exist in the NC client’s folder at the time of bootstrapping as a read-write file.” *Id.*, 6:4-8.

190. Birse’s “server software management process 180 manages access to and from the non-persistent client-specific data 184. For example, at the conclusion

of a user session on an NC client, the server software management process 180 may provide data that is to be preserved between user sessions to the user environment management process 182 and reinitialize the client-specific storage area associated with the NC client for use by the next user.” *Id.*, 6:15-22.

191. Birse’s “user environment management process 182 tracks and maintains the persistent user data 186 to insure that changes the user has made during the current session will be persistent at the next login. For example, upon termination of a user session, preferences, desktop items, and various other information representing changes made by the user during the user session are copies [sic] to a user-specific storage location on the NC server 170.” *Id.*, 6:35-42

192. Thus, “at the user’s next login, from the same or different NC client, the user environment management process 182 will retrieve the data from the corresponding user-specific storage location and return it to the NC client thereby allowing the user to login to any NC client of the NC system 100 and have the same user experience.” *Id.*, 6:43-48.

193. Birse’s “NC client also includes a file system 152, a block device driver 154, a network stack 156, and a network device driver 158, each of which may include hard-wired circuitry or machine-executable instructions or a combination thereof. Furthermore, at least a portion of such hard-wired circuitry and/or machine-executable instructions may be shared between a combination of the file system 152,

the block device driver 154, the network stack 156, and the network device driver 158.” *Id.*, 7:6-14.

194. “The file system 152 represents logical structures and software routines that are used to control access to and from a local storage medium, such as a hard disk system.” *Id.*, 7:21-23.

195. “The block device driver 154 services file system read and write requests to the system volumes 160.” *Id.*, 7:35-35.

196. “The network stack 156 represents logical structures and software routines that are used to control access to and from the data communication link 140.” *Id.*, 7:55-57.

197. Birse discloses an NC client/server system that includes a base/root image (Core System Volume 922) and at least one leaf/incremental snapshot (shadow system volume 930). EX1005 (Birse), 14:24-30, 15:1-7. This is seen in Birse’s Figure 9, reproduced below.

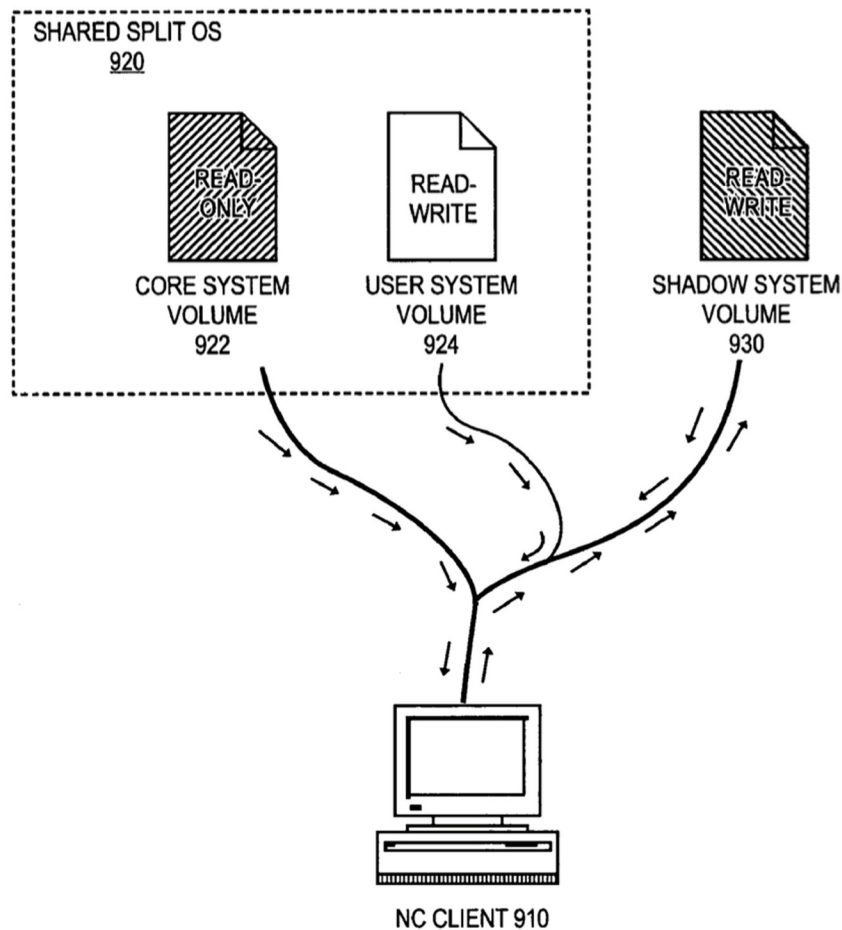


FIG. 9

198. “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.” EX1005 (Birse), 14:24-30.

199. Moreover, Birse states for Figure 9 that “[a]s indicated by the directional arrows along the connections between the core system volume 922, the user system volume 924, and the shadow system volume 930 and the NC client 910, data may be read from each of the core system volume 922, the user system volume 924, and the shadow system volume 930; however, data may only be written to the shadow system volume 930.” EX1005 (Birse), 15:1-7. A POSITA would have understood Birse to manage snapshots as I describe in §VII.D above.

200. In Birse, “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) 60 directed to the user system volume 930 are instead copied to the shadow system volume 930.” EX1005 (Birse), 14:57-61. A POSITA would have understood Birse to manage snapshots using standard CoW techniques. *See also* §VII.D above.

201. “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.” EX1005 (Birse), 7:35-41. A POSITA would have understood Birse to include a (logical/virtual) device driver that resides between file systems and storage devices,

and merges/unifies I/O requests from applications. This functionality is an integral part of snapshot storage systems. *See* §VII.D above.

202. Birse discloses that “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) 60 directed to the user system volume 930 are instead copied to the shadow system volume 930.” EX1005 (Birse), 14:57-61. A POSITA would have understood that Birse’s block driver implements the functionality of managing snapshots including merging their content, using standard CoW techniques. *See also* §VII.D above.

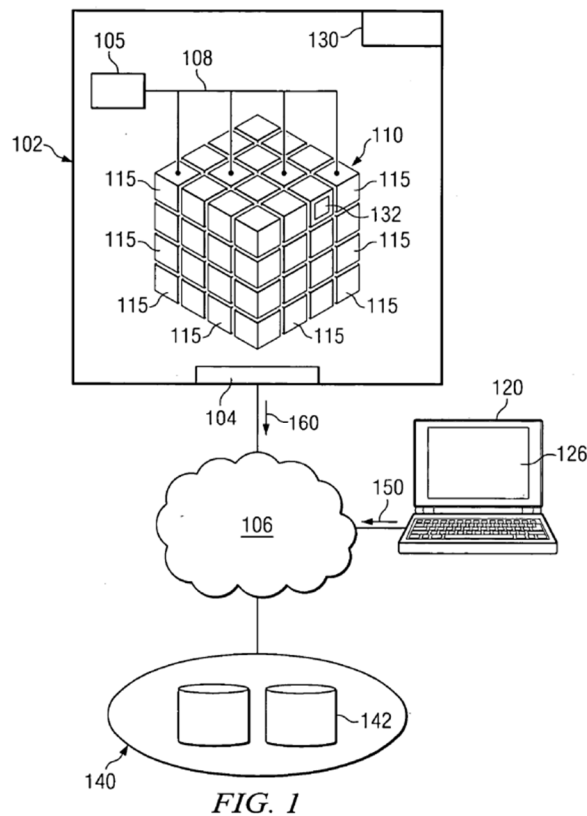
203. **In sum**, Birse discloses data structures and methods that a POSITA would have found functionally identical to the ’282’s UBD, especially in light of the ’282 patent’s description of what a UBD may be and how it reportedly works (*see* §VIII.B above).

B. Ballew

204. Ballew relates to a “high performance computer system and method” in which a “high performance computing (HPC) system 100” is used for “executing software, applications, and processes.” EX1006 (Ballew), Title, [0014]. Ballew’s Background section describes that HPC systems were known and allowed for large tasks to be performed, such as by scientists and engineers for modeling, simulating, and analyzing complex physical or algorithmic phenomena,” and were usually

arranged in the form of “HPC clusters” EX1006 (Ballew), [0002]. The ’282 Patent also discusses using HPC clusters. EX1001 (’282 Patent), 1:20-38 (“[o]ver the years, as the interne[t] has expanded and computers have multiplied, the need for clustered computing such as High Performance Computing (HPC) has increased”), 3:62-4:10.

205. See FIG. 1 of Ballew, below.



206. Ballew’s system 100 offers “HPC functionality dynamically allocated among various computing nodes 115” of an HPC server 102 (multiple servers in a “server pool”), where the plurality of nodes 115 can “comprise any computer or processing device such as, for example, blades, general-purpose personal computers

(PC), Macintoshes, workstations, Unix-based computers, or any other suitable devices.” *Id.*, [0016]-[0017]. The system 100 can operate as a “clustered application server” or a “remote web server.” *Id.* Ballew also discloses that the HPC server 102 can include a grid 110 comprising the plurality of nodes 115, where the grid 110 “is a group of nodes 115 interconnected for increased processing power.” *Id.*, [0016]-[0017]. Ballew’s FIGS. 2C and 2D, shown below, illustrate examples in which a grid 210 has virtual clusters 220 allocated within the grid.

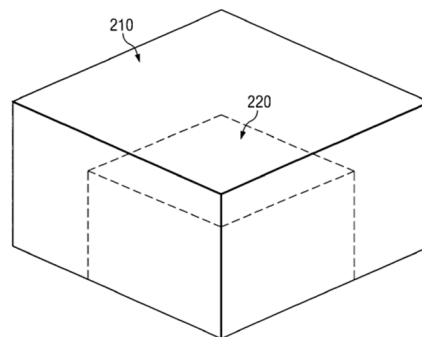


FIG. 2C

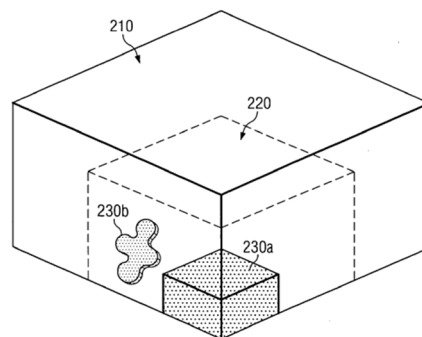


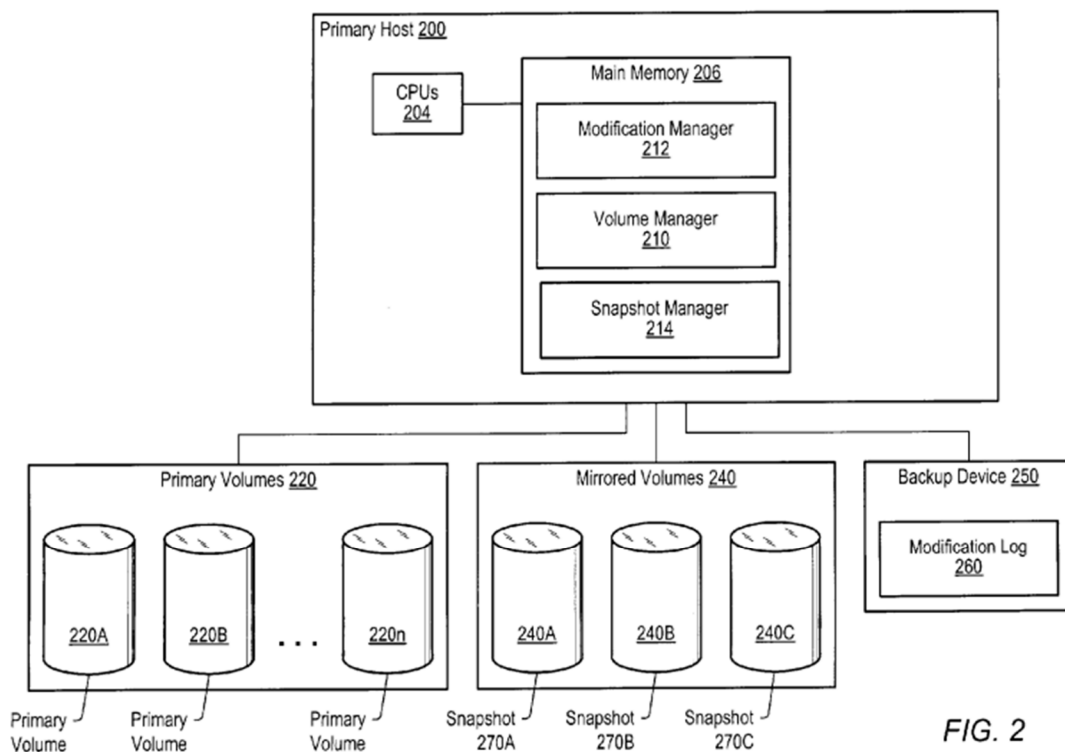
FIG. 2D

207. When the HPC server 102 “is first initialized or booted” a “cluster management engine 130 may dynamically allocate various portions of grid 110 to one or more virtual clusters 220 based on, for example, predetermined policies,” and can “also manage and provide a unique execution environment for each allocated

node of virtual cluster 220.” *Id.*, [0029]. Ballew explains that “[t]he execution environment may consist of the hostname, IP address, operating system, configured services, local and shared file systems, and a set of installed applications and data.”

C. Wu

208. Wu relates to a “techniques for reconstructing deleted snapshot copies of a data volume” in which snapshots “corresponding to an original volume” are created at the “volume-level” by storing data blocks of the original volume in the snapshots. EX1007 (Wu), 1:8-10, 4:51-60. See FIG. 2 of Wu, below.



209. Wu describes embodiments where, “whenever the original volume is modified, a copy of the pre-modification value of the modified block may be stored (e.g., in a copy-on-write file)” and then, “[w]hen the snapshot is used (e.g., to make a backup), the virtual device may be read,” such that “[r]ead accesses to the virtual device targeting a block that was modified subsequent to the creation of the snapshot may be remapped to the stored copy of the pre-modified value of that block” to provide “a point-in-time copy of the original volume.” *Id.*, 4:61-5:4.

210. Wu discloses using a modification log to track modifications to the original volume between any two snapshots, such that, “[e]ach time a modification is detected, modification manager 212 may log the time at which the modification occurred, the address(es) of the modified block(s), and the pre- and post-modification values of the modified block(s).” *Id.*, 6:14-23. In the event that a snapshot is deleted, the modifications stored in the modification log can be used to reconstruct the snapshot using another snapshot. *Id.*, 7:26-29. For example, “[i]f the deleted snapshot was created after the existing snapshot, the snapshot manager 214 may modify the existing snapshot by sequentially applying modifications logged in the modification log(s) 260 between the time the existing snapshot was created and the time the deleted snapshot was created.” *Id.*, 7:36-41.

D. Kleinschnitz

211. Kleinschnitz relates to tools and techniques for facilitating management of storage, software, and other computer resources using system images that can be stored remotely and deployed to the original imaged computer or other computers. EX1008 (Kleinschnitz), Abstract, 1:8-10, 4:51-60. See FIG. 2 of Kleinschnitz, below.

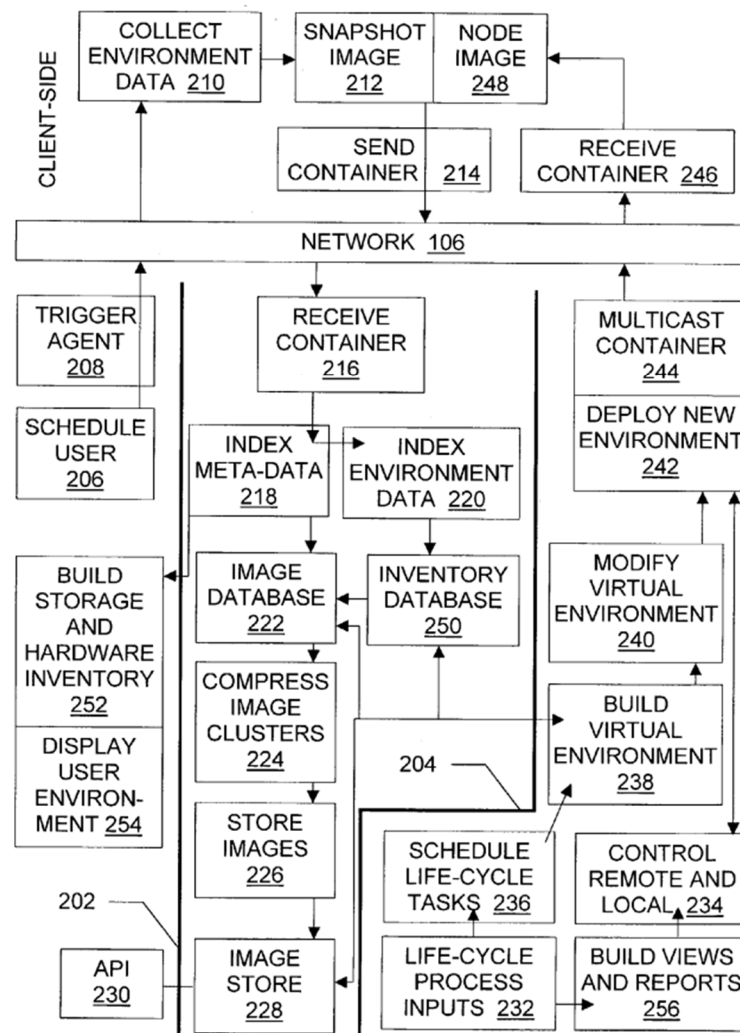


Fig. 2

212. Kleinschnitz discloses using snapshots that include “incremental images” reflecting “the state of the computer’s storage” and storing the snapshots at another storage location. *Id.*, [0003], [0035], [0059]. Kleinschnitz explains that “an incremental image 104 may be merged with a prior complete snapshot image 104 to form a complete and more recent snapshot image.” *Id.*, [0071]. Kleinschnitz also discloses that storage disks can be partitioned, such as “by placing user data in one partition and system data and software in another partition.” *Id.*, [0079].

XI. CLAIM CONSTRUCTION

213. I understand that, for purposes of my analysis in this *inter partes* review proceeding, the terms appearing in the patent claims should be interpreted according to their “ordinary and customary meaning.” In determining the ordinary and customary meaning, I understand that the words of a claim are first given their plain meaning that those words would have had to a person of ordinary skill in the art at the time of the alleged invention. I also understand that the structure of the claims, the specification, and file history may be used to better construe a claim. Moreover, I understand that even treatises and dictionaries may be used, albeit under limited circumstances, to determine the meaning attributed by a person of ordinary skill in the art to a claim term.

214. As detailed in my analysis below, the challenged claims are obvious under any reasonable interpretation of the claim language.

215. For purposes of this declaration, I've been asked to assume that the time of the alleged invention is December 30, 2004 (the filing date of U.S. Patent Application 11/026,622, EX1009). However, my opinions would not change even if using the actual filing date of the '282 Patent of March 30, 2006.

XII. DETAILED EXPLANATION OF GROUNDS

A. Ground 1: Birse Renders Claim 1, 3-10, 13-16, 19-26, 29, 30 Obvious.

1. Claim 1

a. [1.0] A system for distributing an application environment comprising:

216. To the extent the preamble is limiting, Birse discloses a system and method for distributing an application environment (from NC server 170) to a plurality of compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826): "...[T]he NC clients [150] boot over the network by accessing a stored copy of the operating system from the NC server [170]. Additionally, when NC clients want to run an application, they access it from the NC server." EX1005 (Birse), 4:56-60. *See also id.*, 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications). This is illustrated in Birse's Figure 1, reproduced below:

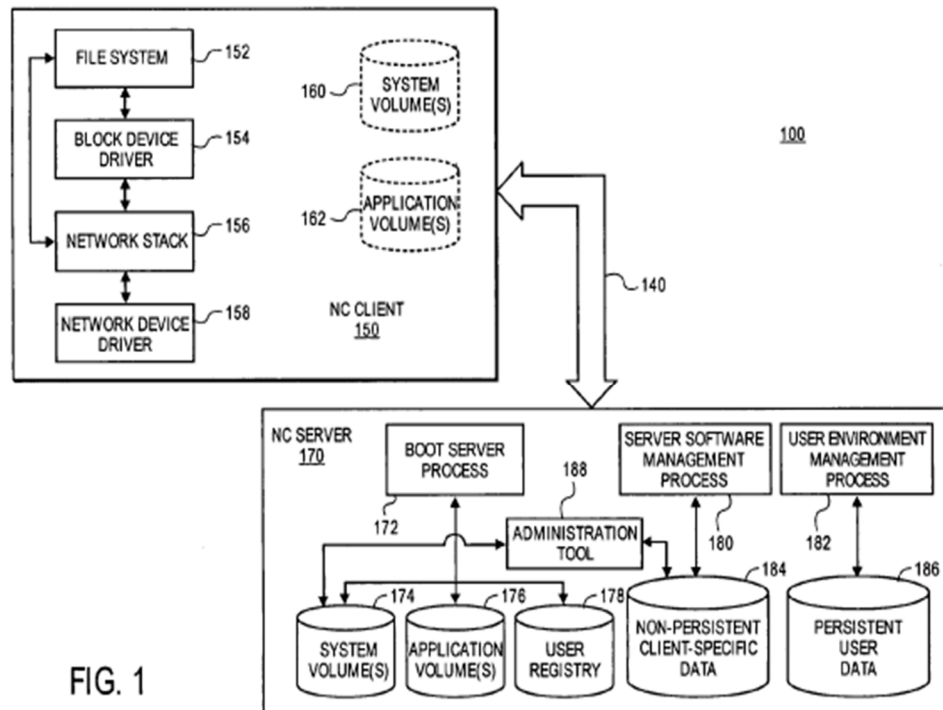


FIG. 1

217. **In sum**, Birse discloses a system [Birse's system 100] for distributing an application environment [Birse using stored operating system volumes].

b. [1.1] a compute node comprising a computer system; a first storage unit for storing blocks of a root image of the compute node,

218. Birse relates to an "NC system" that includes "one or more NC clients," (*i.e.*, "a compute node"), and "one or more NC servers" that "may provide access to a copy of the operating system to an NC client in response to a boot request." EX1005 (Birse), 4:31–40. "FIG. 8 is a simplified block diagram illustrating two NC systems 810 and 820 coupled via the Internet 805." *Id.*, 14:8-9. Birse's Figure 8 is reproduced below:

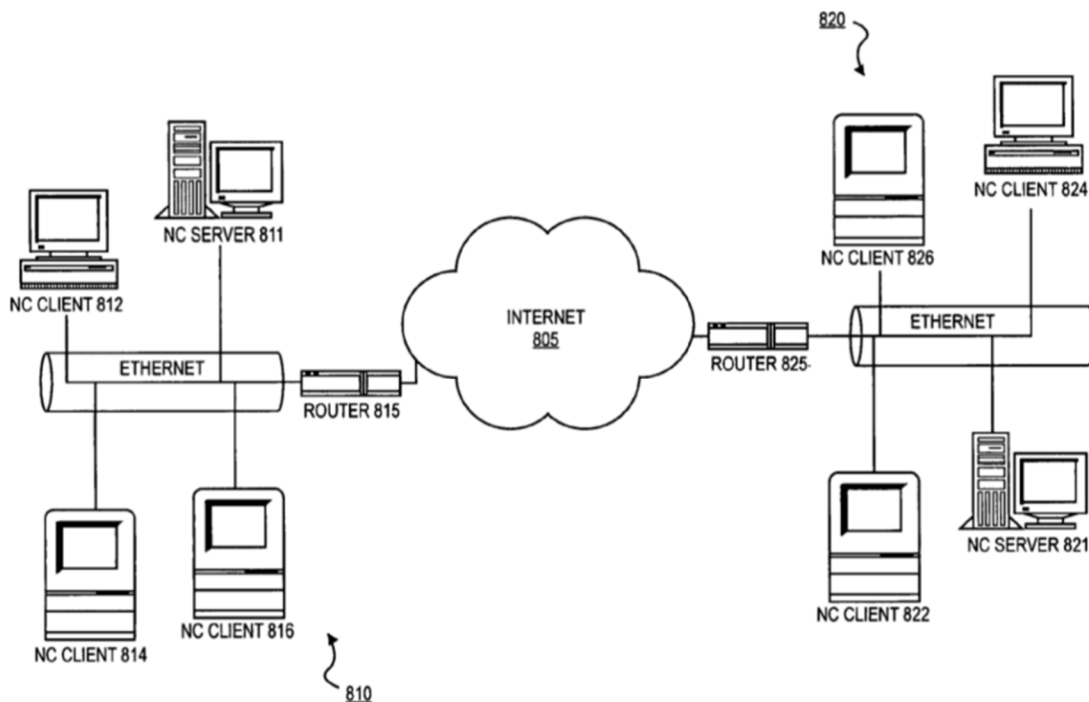


FIG. 8

219. Moreover, NC server 170 and NC clients 150 are “network computer[s] (NC)” (*id.*, 1:57-58) and are illustrated by exemplary digital processing system 200 of FIG. 2. As shown in FIG. 2 of Birse and described at 7:63-8:35 of Birse, the compute nodes comprise a computer system (“the digital processing system 200 shown in FIG. 2 may be used as an NC client, an NC server, or other server *computer system*”).

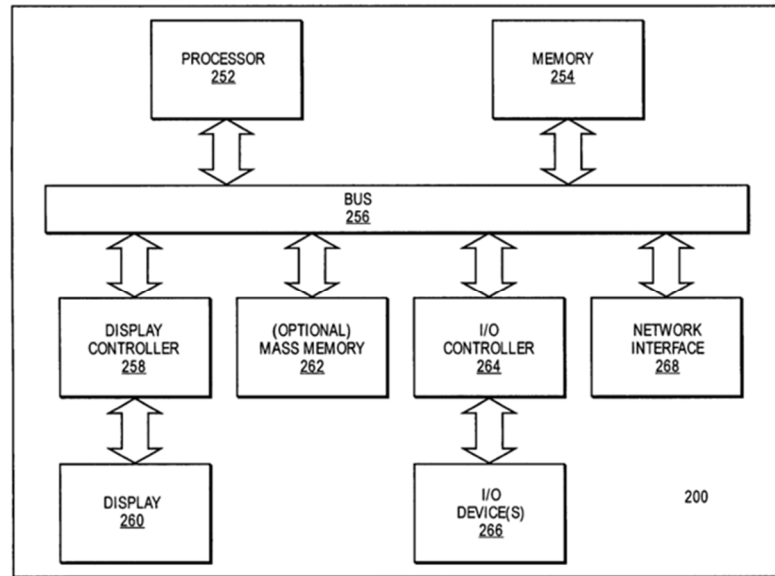


FIG. 2

220. Birse further discloses storing blocks of a root image (system volumes 174, 650, 660, 922, 924 on NC server 170) of a compute node on a first storage unit (one or more hard drives 610, logically separate in NC folders/directories 630, 640): “The NC server 170 includes one or more system volumes 174.... The system volumes 174 include a protected, read-only, master copy of the operating system software.” EX1005 (Birse), 5:10-14.

221. Furthermore, “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.... 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.” *Id.*, 14:27-34, 14:41-44.

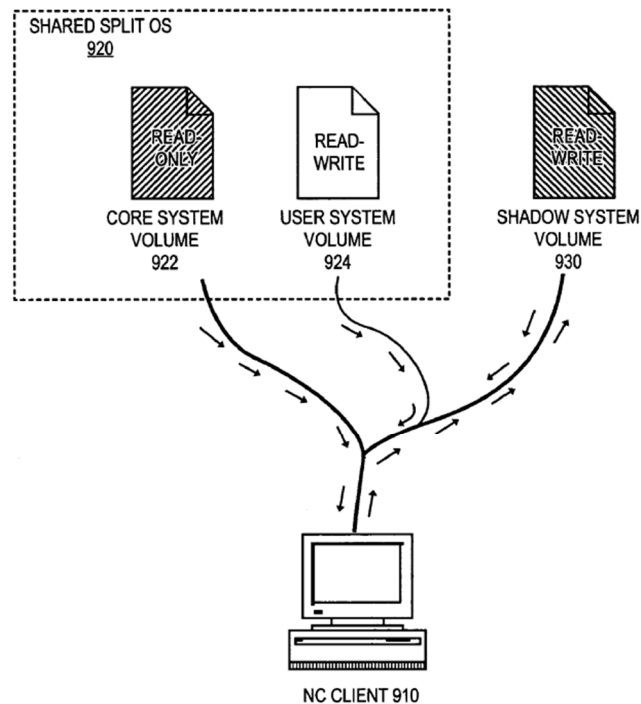


FIG. 9

222. As shown above by the arrows of FIG. 9, and as further explained below, the NC client reads data from the core system volume 922 and user system volume 924 and writes requested by the NC client are directed to the shadow system volume 930. EX1005 (Birse), 14:56-67. The “system volumes 174 ... may be independently distributed across multiple server computer systems.” *Id.*, 6:59-63.

223. FIG. 6 of Birse “is a simplified block diagram illustrating the layout of an exemplary hierarchical directory structure that may be used by an NC server [170] according to one embodiment.... [A] hard drive 610 of 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.” *Id.*, 12:30-38. Birse’s Figure 6 is reproduced below:

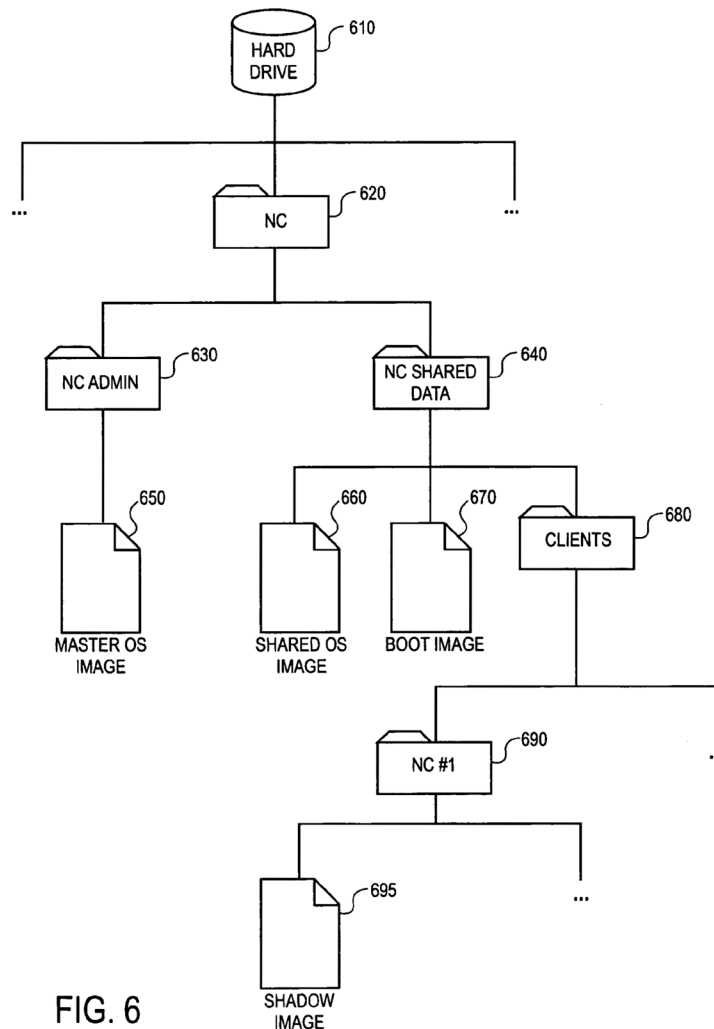
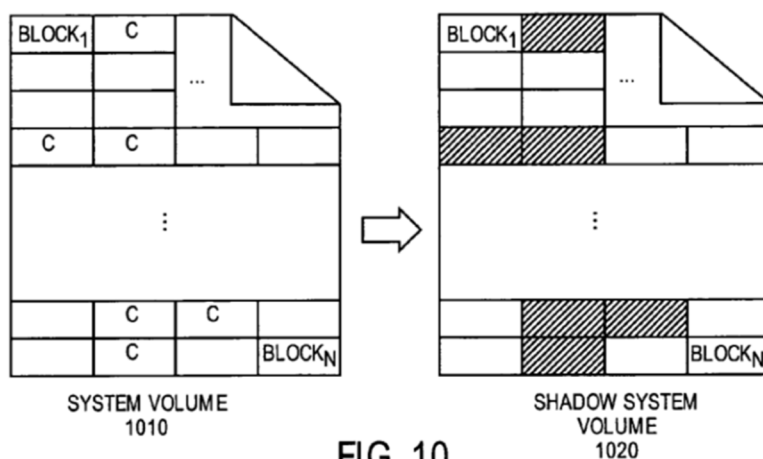


FIG. 6

224. As shown in FIG. 6, Birse discloses storing blocks of the root image on a first storage unit (specific folders/directories on hard drive 610). That is, Birse

discloses storing in an NC folder 620 (a first storage unit) on hard drive 610 the master (read-only) OS image 650 (in folder 630) and the shared OS image 660 (in folder 640). “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.” *Id.*, 12:44–47.

225. System volumes (a.k.a. system volumes 174, 650, 660, 922, 924, 1010 shown in FIGS. 1, 9, and 10) are stored as blocks as shown in Fig. 10, reproduced below:



226. Additionally, although Birse discloses using both a master OS image 650 / user system volume 922 and a shared OS image 660 / user system volume 924, it would be apparent to a POSITA that these components meet the claimed root image of the '282 Patent. In Birse, the shared OS image 660 / user system volume 924 is located in a storage location accessible to the Birse's block device driver 154 and is simply “a read-write version of the read-only master operating system image

650” to ensure that the original master OS image 650 is stored securely and to avoid any attempt to “delete or move items that are essential to system operation.” EX1005 (Birse), 14:35-40. However, a POSITA would have found it obvious that the master OS image 650 / user system volume 922 and the shared OS image 660 / user system volume 924 operate in the same way as the claimed root image to store the original, unmodified, blocks of the system volume because an OS (kernel) image alone is only part of what a computer system needs to run: it also needs a system volume with an assortment of critical administrator/user utilities (*e.g.*, to browse folders, read configuration files, configure a network, manage user accounts, and many more).

227. **In sum**, Birse discloses a compute node [Birse’s NC client 150] comprising a computer system [Birse’s digital processing system 200] and a first storage unit [Birse’s NC admin/shared directories on hard drive 610] for storing blocks of a root image [Birse’s system volumes 174, 650, 660, 922, 924, 1010] of the compute node.

- c. **[1.2] wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;**

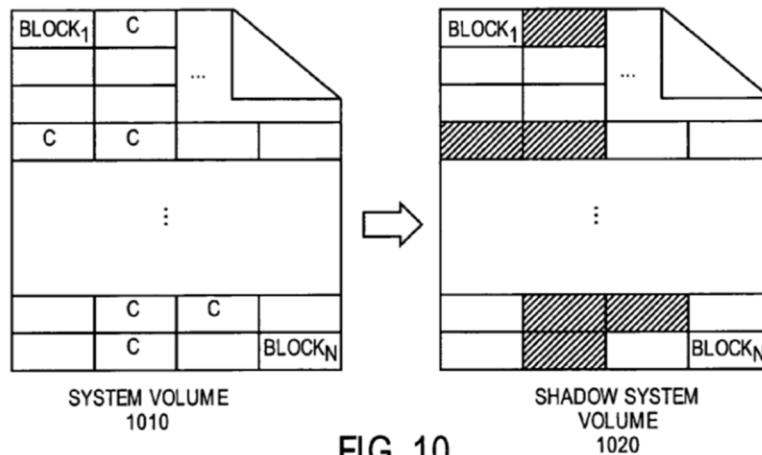
228. Birse discloses that the first storage unit is a first non-volatile memory. As described above, the system volume in Birse (the root image) is stored within a

folder/directory on a “hard drive 610 ... or other mass storage device.” EX1005 (Birse), 34-42. A POSITA would understand that a hard drive or other mass storage device is non-volatile memory. *See* §VII.A above.

229. Indeed, Birse specifically discloses that hard drives/disks are non-volatile memory: “a non-volatile mass memory (e.g., a hard disk).” EX1005 (Birse), 16:1-7.

230. Birse also discloses that the root image comprises a computer program. As described above, Birse specifically discloses that the system volumes comprise an operating system (e.g., operating system images 650, 660). EX1005 (Birse), 12:44-47. Birse specifically discloses that “[a]n ‘operating system’ refers to a **program** that manages a computers internal functions and provides a means to control the computer’s operations and file system.” EX1005 (Birse), 4:7-9.

231. Birse also discloses that the blocks of the root image comprise sections of data. First, it is noted that the word “section” only appears in the claims of the ’282 Patent. As explained above, Birse’s system volumes (a.k.a. system volumes 174, 650, 660, 922, 924, 1010) are stored as blocks (sections of data) as shown in Fig. 10, reproduced below:



232. Birse further describes embodiments in which data can be written in bands, where bands are a predetermined number of blocks. These bands of blocks include sections of data, *e.g.*, bands of “128K (256 512-byte blocks).” EX1005 (Birse), 15:20-32.

233. Birse further discloses or suggests that a file comprises at least one block. As described above, the system volumes are operating system images. Operating systems include files and it would be obvious that, depending on the size of a file of the operating system stored in the operating system volumes of Birse, that one file could comprise at least one block of the system volume. As explained in §VII.D above, it was well-known prior to the '282 Patent that the content of a file can be stored in one or more blocks or sectors. *See, e.g.*, EX1017 (Le), 1:16-20.

234. **In sum**, Birse discloses wherein the first storage unit comprises a first non-volatile memory [Birse’s hard disk 610], wherein the root image [Birse’s system volumes 174, 650, 660, 922, 924, 1010] comprises a computer program, wherein the

blocks comprise sections of data [Birse's data blocks of system volumes 174, 650, 660, 922, 924, 1010], and wherein a file of the root image comprises at least one block [Birse's blocks of operating system volumes 174, 650, 660, 922, 924, 1010].

- d. **[1.3] a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, wherein the second storage unit comprises a second non-volatile memory; and**

235. As explained in §X.A above, Birse discloses using a copy-on-write (CoW) technique to store a leaf image (shadow volume/image 930, 695) for respective compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826) on a second storage unit (NC client folders 690 #1...n on hard drive 610), where the leaf image includes new data blocks of the root image (write requests for new data) and changes to the blocks of the root image (write request for changes to operating system data): “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.” EX1005 (Birse), 12:54–59. This is shown in Birse's FIG. 6, reproduced below:

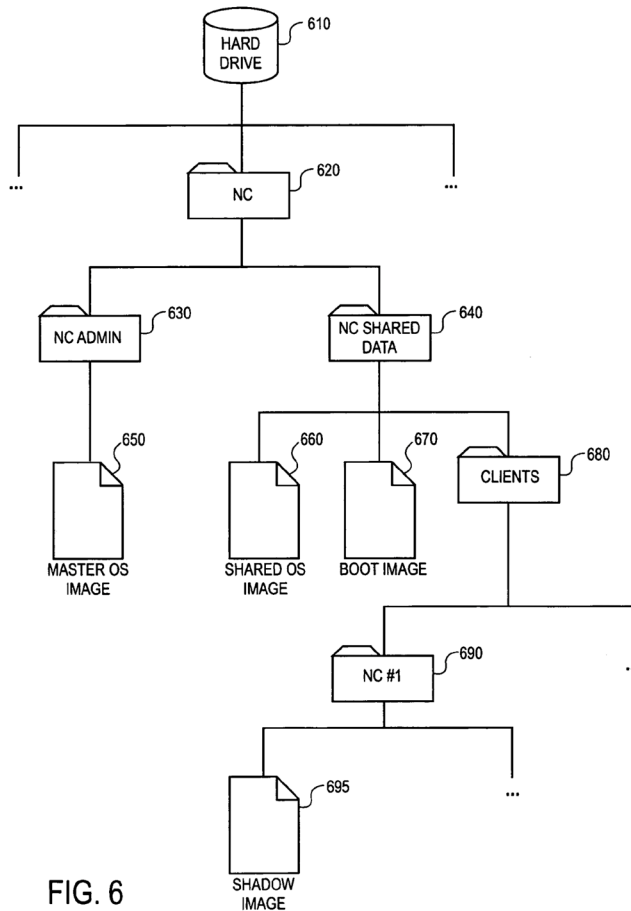


FIG. 6

236. Birse discloses that the second storage unit comprises a second non-volatile memory. First, the '282 Patent describes that “[t]he first storage unit 140 and second storage units 150a-n may each be contained on separate physical storage devices, on separate logical spaces on the same storage device, or any combination thereof.” EX1001 ('282 Patent), 4:17-22. In embodiments of Birse, the first storage unit and the second storage units are in separate logical spaces on the same storage device, that is, the first storage unit is in NC folders/directories 630, 640 (containing the master OS image and shared OS image (root image)) and the second storage unit

is in NC folders/directories 680, 690 (containing the shadow image (leaf image)). EX1005 (Birse), 12:30-64.

237. To the extent that Patent Owner asserts that the first non-volatile memory and the second non-volatile memory must be separate physical storage devices (*e.g.*, separate hard drives), it would be obvious to do so based on the disclosure of Birse. Particularly, Birse discloses that I/O device 266 shown in FIG. 2 can include multiple “disk drives” (*id.*, 8:32-35), and that memory used for storage can naturally “comprise one or more disks” (*id.*, 15:58-62). In addition, while Birse describes “a single NC server 170” providing server processes and data including the system volumes 174 and client-specific data 184 (where the client-specific data 184 can include shadow volumes (leaf images)), Birse also discloses that data can be “distributed across multiple server computer systems.” *Id.*, 6:59-63, 7:41-46.

238. Regarding the leaf image, Birse discloses that “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 ... [but is] 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 [695], 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.” EX1005 (Birse), 7:35-53 (emphasis added); *see also id.*, 11:30-32.

239. FIG. 4 of Birse “is a flow diagram illustrating how NC client [150] write requests directed to the system volumes 160 are handled.... [A]t 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.” *Id.*, 10:63-65, 11:6-17.

Birse’s Figure 4 is reproduced below:

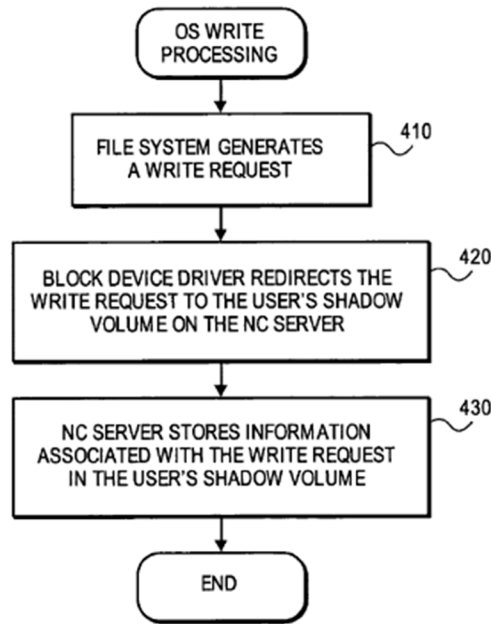


FIG. 4

240. Embodiments of Birse disclose 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.” *Id.*, 11:49-52. Birse also states that “the shadow image 695 preferably contains only portions of the shared operation system image 660 that have been modified by the user.” *Id.*, 12:59-61.

241. Similarly, “at step 510, the file system 152 generates a read request directed to the system volumes 160,” which “is received by the block device driver 154” at step 520. This is seen in Birse’s FIG. 5, reproduce below:

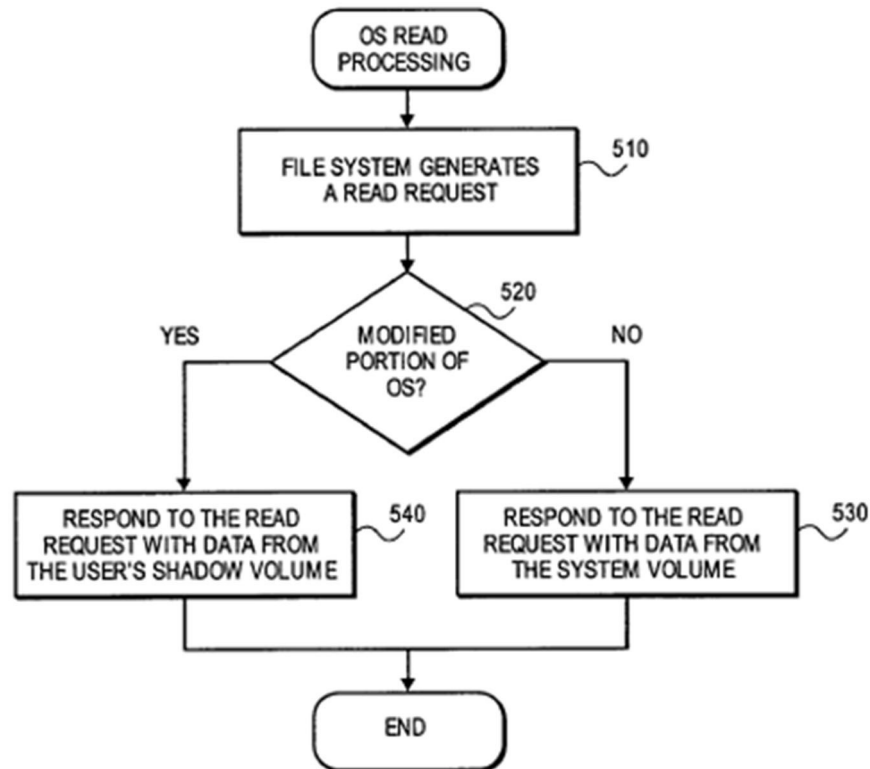


FIG. 5

242. Rather than reading directly from the system volume, “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.” *Id.*, 11:55–62. 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,” wherein “the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174.” *Id.*, 11:62–12:3. Alternatively, “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,” wherein “the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user’s shadow volume.” *Id.*, 12:4–6.

243. The NC server “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.” *Id.*, 14:49-53.

244. Moreover, “... 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.” *Id.*, 14:56-61.

245. These modifications in the shadow system volumes of Birse are stored as blocks, as shown for example in Fig. 10, reproduced below:

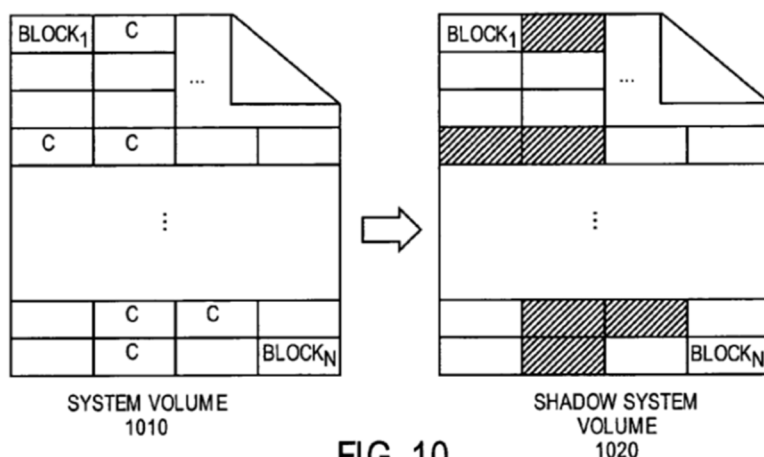


FIG. 10

246. Birse further describes embodiments in which data can be written in bands in a sparse shadow system volume, as shown in FIG. 11 below.

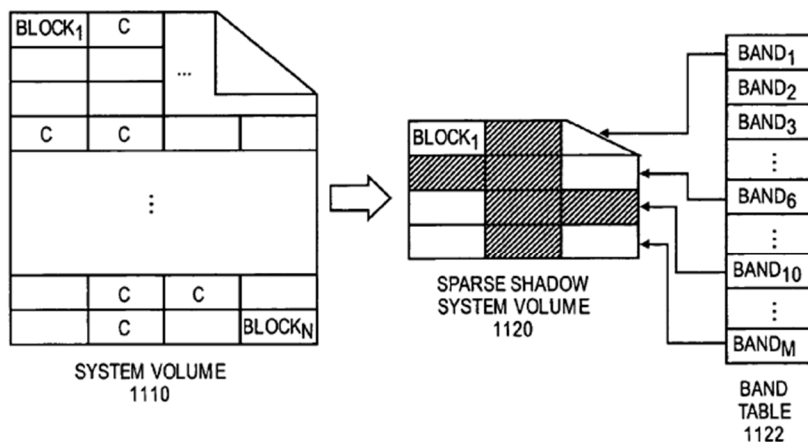


FIG. 11

EX1005 (Birse), 15:20-32.

247. The sparse shadow system volume 1120 “is written in band increments,” where a band “is a predetermined number of blocks,” such that “the disk space used by the shadow volume 1120 will consist only of the actual data written (in band increments).” *Id.*

248. This banding feature provides an “inherent one-to-one block mapping in copy-on-write images” which Birse discloses can also “be replaced with a table or other data structure that maps a band of the primary file to a corresponding band in the shadow file.” EX1005 (Birse), 15:32-37. Logic may be “included in the block device driver 154, for example, to break up requests into the largest contiguous

chunks possible” so that “both reads and writes can span non-contiguous band boundaries.” EX1005 (Birse), 15:37-40.

249. Banding as described in Birse “reduce[s] disk space requirements of the NC client shadow system images” and “the reduction is achieved by only storing those bands which have actually been written to. EX1005 (Birse), 15:41-45. Birse describes that the “an exemplary band table 1122 is a simple array allocated at initialization” and entries in the table are assigned BAND_NONE until the corresponding band is written to, at which point the next available band in the shadow file is assigned.” EX1005 (Birse), 15:45-49. Lastly, “in order to encourage contiguous blocks on the NC server’s file system, in one embodiment, if bands are allocated during a write, the shadow file size is increased to the appropriate number of bands before the write request is sent.” EX1005 (Birse), 15:45-52.

250. **In sum**, Birse discloses a second storage unit [Birse’s NC client directory 690 on hard disk 610] for storing a leaf image [Birse’s shadow volumes 695 and 930], the leaf image comprising new data blocks and changes to the blocks of the root image [Birse’s blocks of shadow volumes include blocks changed from root image], wherein the second storage unit comprises a second non-volatile memory [Birse’s hard disk 610].

- e. **[1.4] a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node, wherein the union block device comprises a driver,**

251. Birse teaches the use of a block device *driver* that redirects read and write requests to/from a user's shadow volume as necessary. EX1005 (Birse), 7:35-53. Accordingly, it would have been obvious to a POSITA that the block device driver for each NC client is "a union block device for interfacing between the compute node and the first and second storage units ... wherein the union block device comprises a driver," as required by this claim." *See also* §VIII.B above.

252. Furthermore, the block device driver, system volumes, and shadow volumes "distribute the application environment to the compute node" as required by claim 1, and thus also "deliver[] the application environment to the compute node," as required by claim 15 (§XII.A.12.f below). The '282 patent explains that, "[i]n a branching store file system, a read-only base image (or 'root' image) of the application environment is created," which is "accessible by all compute nodes in the cluster," and that "[c]hanges made by a compute node to the root image are stored in a 'leaf' image unique to that compute node.'" EX1001 ('282 Patent), 1:62-67.

253. The Birse system operates the same way: the system volumes include "a protected, read-only, master copy of the operating system software," and a "shadow volume contains only those portions of the operating system that have

been modified by the user.” EX1005 (Birse), 5:13–14, 11:49–52. The block device driver redirects read and write requests between the system volumes and shadow volumes in a way that is transparent to the file system. In one embodiment, “the block device driver 154 appears to the file system 152 to be a standard hard drive device driver,” and “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.” *Id.*, 7:35–38, 11:14–17.

254. **In sum**, Birse discloses a union block device [Birse’s block device driver 154] for interfacing between the compute node and the first and second storage units [Birse’s NC client 150, NC admin/shared directories on hard drive 610, and NC client directory 690 on hard disk 610] to distribute the application environment to the compute node [Birse’s NC client 150], wherein the union block device comprises a driver [Birse’s block device driver 154].

- f. [1.5] wherein the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit;**

255. The ’282 patent explains that “[a] filter operates between the compute nodes and the file system(s), which merges the changes recorded on the leaf images with the root images and delivers the result to the appropriate compute node,” which

appears to a compute node as “its own unique and cohesive instance of the application environment.” *Id.*, 1:67-2:5.

256. The Birse system operates the same way: the block device drivers 154 of Birse (*i.e.*, “union block devices”) “are configured to create said application environments by merging the blocks of said root image,” that is, data from the system volumes, “with the blocks of respective leaf images,” that is, the shadow volumes.

257. Birse discloses merging the blocks of the root image stored on the first storage unit (blocks of the operating system software stored on system volumes 174, 650, 660, 922, and 924) with the blocks of the leaf image (695 and 930) stored on the second storage unit (NC client folders 690 #1...n) to create the application environment (dynamically merged system environment): “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.” EX1005 (Birse), 14:61-67.

258. When “an NC client [150] boots from the network and accesses the operating system from an NC server [170], the user’s preferences are dynamically merged with the system environment provided to the NC client [150].” *Id.*, 9:2-5.

Also, “[a]dvantageously, since the user’s desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client [150] as it boots from the network, the user may login to any NC client [150] on the network and have the same user experience.” *Id.*, 9:5-10.

259. “Referring now to Fig. 5, NC client [150] 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.” Thus, “[i]n 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.” *Id.*, 11:39-41, 11:55-12:10. Birse’s Figure 5 is reproduced below:

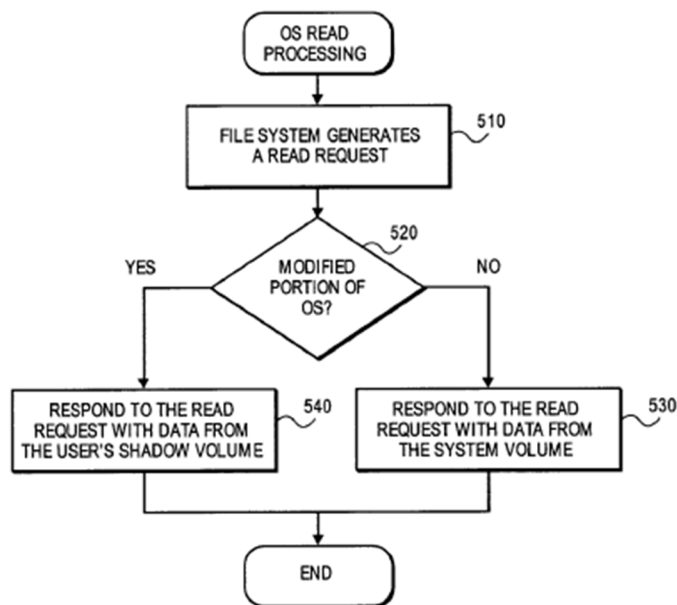
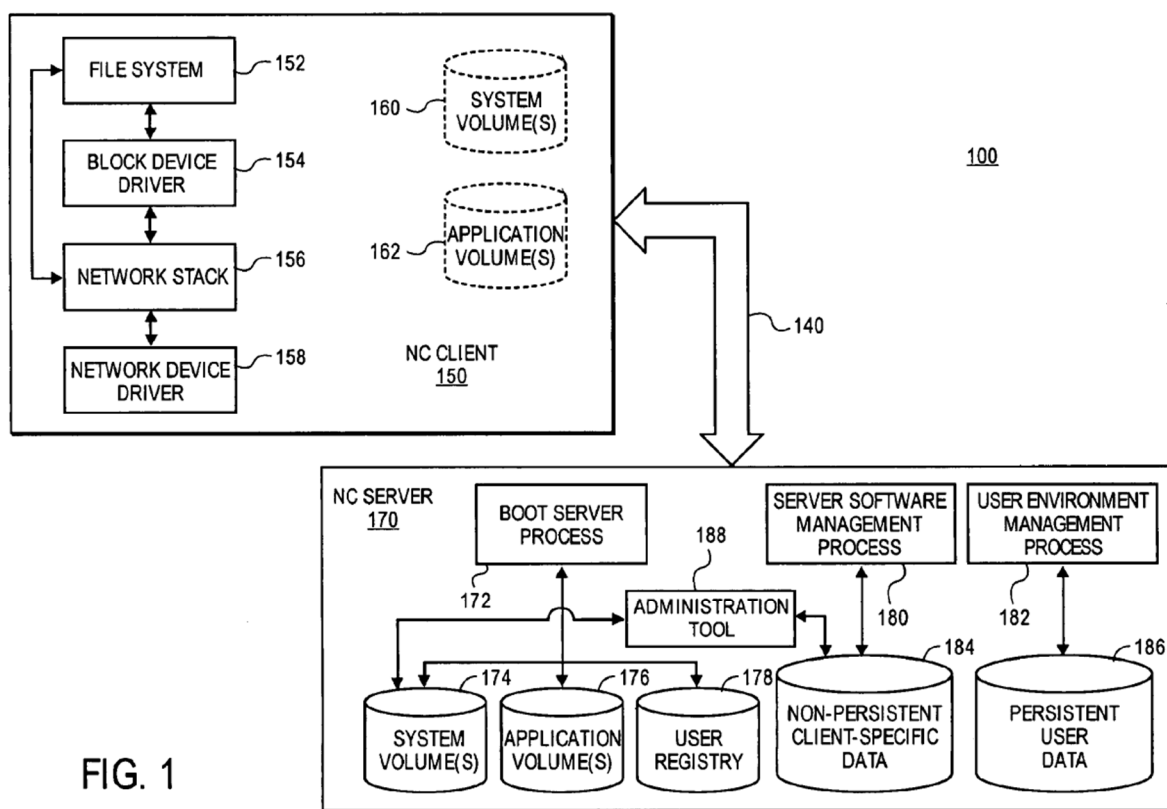


FIG. 5

260. **In sum**, Birse discloses the union block device [Birse’s block device driver 154] creates the application environment [Birse’s stored operating system volumes] by merging [Birse’s dynamic merging of user data with the system environment] the blocks of the root image [blocks of Birse’s system volumes 174, 650, 660, 922, 924, 1010] stored on the first storage unit [Birse’s NC admin/shared directories on hard drive 610] with the blocks of the leaf image [blocks of Birse’s shadow volumes 695 and 930] stored on the second storage unit [Birse’s NC client directory 690 on hard disk 610].

- g. [1.6] the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node;

261. As described above, Birse teaches the use of a block device *driver* that “keep[s] track of which portions of the operating system ... have been modified by the user” and redirects file system read and write requests between the system volumes and shadow volumes as appropriate. EX1005 (Birse), 12:11-15. See FIG. 1 of Birse:



262. The “block device driver 154 appears to the file system 152 to be a standard hard drive device driver.” *Id.*, 7:35–38. Additionally, Birse specifies that

the file system and block device driver operates “as if the contents of the system volumes 160 are, in fact, contained on a local hard drive and the file systems read and write requests are redirected at a *lower level* of the operating system.” *Id.*, 7:27-31.

263. **In sum**, Birse discloses the union block device [Birse’s block device driver 154] comprises a low-level driver [Birse’s block device driver 154 redirects read and write requests at a lower level of the operating system] for interfacing between the first and second storage units [Birse’s NC admin/shared folders NC client folders on hard disk 610] and the file system [Birse’s file system 152] of the compute node [Birse’s NC client 150].

h. [1.7] and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X.

264. As an initial matter, a POSITA would have understood that any storage system that maintains files, executables, CoW snapshots, and especially an operating system image, *must* persistently preserve the data blocks associated with those. In other words, data is persistent on non-volatile devices (*e.g.*, hard disks). If data was not persisted, for example maintained only in volatile RAM memory, all data (*e.g.*, user files and operating system images) would be lost upon system reboot or power outage. That would render such computer systems non-functional and useless. Additionally, for any storage snapshot system or system-imaging system that stores

snapshots or system images persistently, there must be a mapping of the original vs. modified blocks or it would not be possible to collapse/merge, remove, take, and/or restore snapshots or images. *See* §§VII.A above and VII.B above. Furthermore, as explained above in §VII.D above, it was well-known prior to the '282 Patent that sectors and blocks refer to the same subject matter and that sector-based image formats describe the state of a disk at the sector, or block, level. *See, e.g.,* EX1017 (Le), 1:16-20.

265. The only description associated with this claim language in the '282 Patent states:

UBDs 130*a-n* may also modify the leaf image in response to their respective compute node's access to its instance of the application environment. For example, upon receiving a write request from their respective compute nodes for a sector X, the UBDs 130*a-n* will create an appropriate persistent mapping for sector X and then write sector X onto their respective second storage units 150*a-n*, where sector X can then be modified. It will be appreciated that the data block being modified may already exist in the leaf image, in which case it does not need to be mapped and copied from the root image before modification. EX1001 ('282 Patent), 5:1-11.

266. The system of Birse meets these claim elements. In Birse, “one or more shadow volumes [695], may be created for the NC client 150.” 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.” EX1005 (Birse), 7:35-53; *see also id.*, 11:30-32. Birse further explains that “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,” and “the NC server 170 stores information associated with the write request in the shadow volume associated with the NC client 150.” *Id.*, 10:63-65, 11:6-17. As shown for example in FIG. 10 of Birse, blocks of the shadow volume (leaf image) that are modified in response to a write request (a sector X of the shadow volume) are persistently mapped (a “one-to-one block mapping in copy-on-write images”) and correspond to associated blocks of the original system volume (root image).

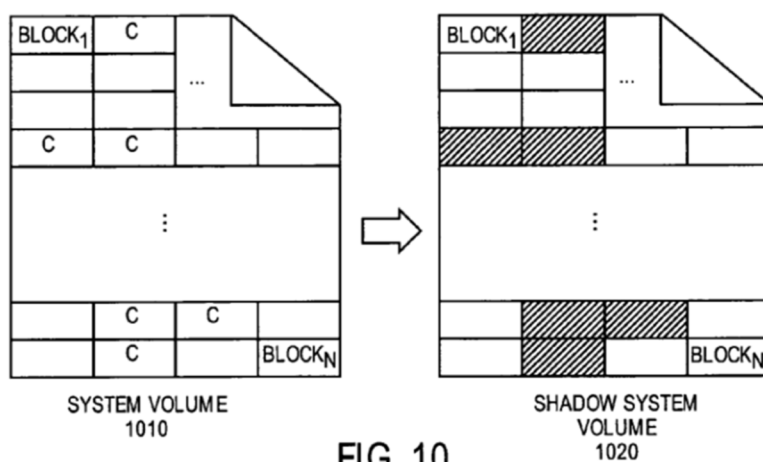


FIG. 10

267. Birse also discloses an embodiment in which a “table or other data structure ... maps a band of the primary file to a corresponding band in the shadow

file.” *Id.*, 10:63-65, 11:6-17, 15:33-37. Specifically, Birse describes embodiments in which data can be written in bands in a sparse shadow system volume, as shown in FIG. 11 below.

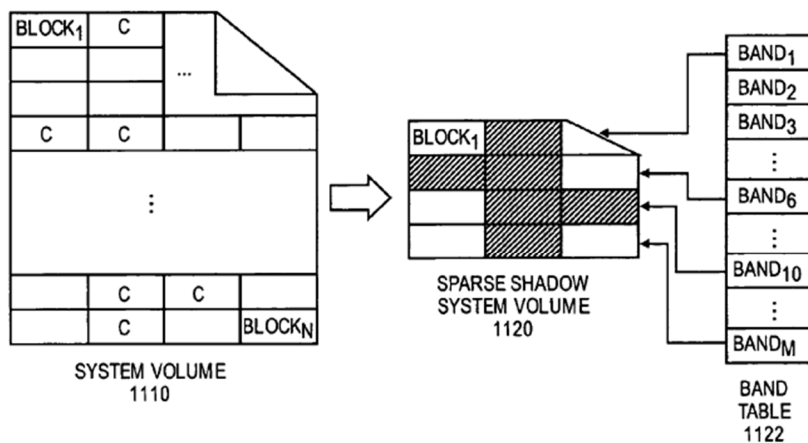


FIG. 11

EX1005 (Birse), 15:20-32.

268. The sparse shadow system volume 1120 “is written in band increments,” where a band “is a predetermined number of blocks,” such that “the disk space used by the shadow volume 1120 will consist only of the actual data written (in band increments).” *Id.*

269. This banding feature provides an “inherent one-to-one block mapping in copy-on-write images” which Birse discloses can also “be replaced with a table or other data structure that maps a band of the primary file to a corresponding band in the shadow file.” EX1005 (Birse), 15:32-37. Logic may be “included in the block device driver 154, for example, to break up requests into the largest contiguous

chunks possible” so that “both reads and writes can span non-contiguous band boundaries.” EX1005 (Birse), 15:37-40.

270. Banding as described in Birse “reduce[s] disk space requirements of the NC client shadow system images” and “the reduction is achieved by only storing those bands which have actually been written to. EX1005 (Birse), 15:41-45. Birse describes that “an exemplary band table 1122 is a simple array allocated at initialization” and entries in the table are assigned BAND_NONE until the corresponding band is written to, at which point the next available band in the shadow file is assigned.” EX1005 (Birse), 15:45-49. Lastly, “in order to encourage contiguous blocks on the NC server’s file system, in one embodiment, if bands are allocated during a write, the shadow file size is increased to the appropriate number of bands before the write request is sent.” EX1005 (Birse), 15:45-52.

271. As explained in §VII.D above, it was well-known prior to the ’282 Patent to use sparse image files like that in Birse to provide a sector-level representation of a volume or disk. *See, e.g.*, EX1017 (Le), 3:1-12.

272. **In sum**, Birse discloses the union block device [Birse’s block device driver 154], upon receiving a write request from the compute node for a sector X [Birse’s block device driver 154 redirects write requests], creates an appropriate persistent mapping for sector X [Birse’s block device driver 154 maps data blocks of write requests to blocks of shadow volumes].

2. Claim 3: The system as recited in claim 1 wherein the compute node comprises a thin-client workstation.

273. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

274. Birse discloses that its NC clients can be “thin clients” that “generally refers to a disk-less computer system that relies upon external means, such as a server, to supply it with an operating system and application programs.” EX1005 (Birse), 4:27-35. A POSITA would have understood that Birse’s disclosure of its NC clients as “thin clients” is equivalent to a thin-client workstation as described in the ’282 Patent.

275. **In sum**, Birse discloses wherein the compute node comprises a thin-client workstation [Birse’s NC clients 150].

3. Claim 4: The system as recited in claim 1 wherein the root image comprises an operating system.

276. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

277. As I explained regarding the independent claims, Birse specifically discloses that the system volumes comprise an operating system (*e.g.*, operating system images 650, 660). EX1005 (Birse), 12:44–47. Birse specifically discloses that “[a]n ‘operating system’ refers to a *program* that manages a computer’s internal

functions and provides a means to control the computer's operations and file system.” EX1005 (Birse), 4:7-9.

278. **In sum**, Birse discloses wherein the root image comprises an operating system [Birse's operating system images 650, 660].

4. Claim 5: The system as recited in claim 1 wherein the root image is concurrently accessible to a plurality of compute nodes.

279. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

280. The Birse system can include a plurality of compute nodes (NC clients) as shown, for example, in FIG. 8 below.

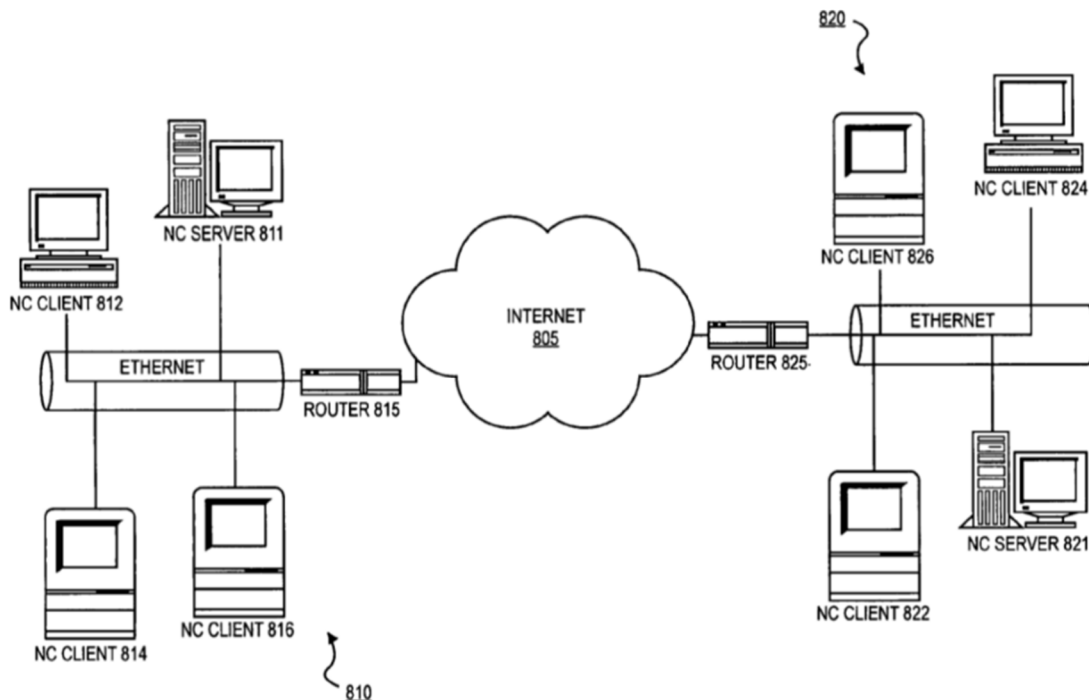


FIG. 8

281. The OS system volume (root image) is made concurrently accessible to the NC clients by the NC server “that services requests of NC clients” including “provid[ing] access to a copy of the operating system ... in response to a boot request.” EX1005 (Birse), 4:31-40. Thus, “when NC clients want to run an application, they access it from the NC server.” EX1005 (Birse), 4:59-60. As explained regarding the independent claims, the system volume (root image) is provided in a location that is accessible to all NC clients (NC shared data folder 840), with NC client specific data (*e.g.*, shadow volumes (leaf images)) being stored in NC client specific locations like folder 690, as shown in FIG. 6 below.

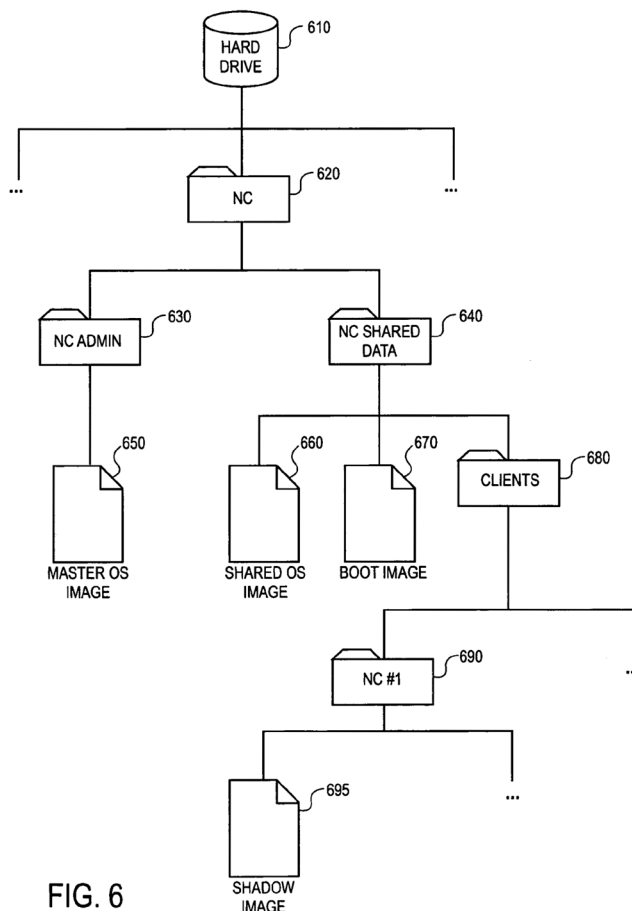


FIG. 6

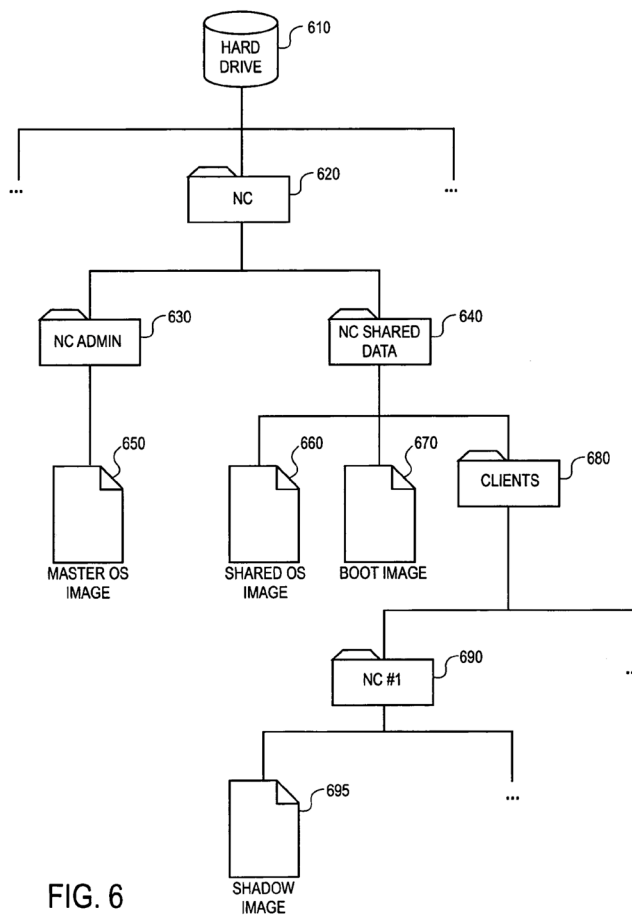
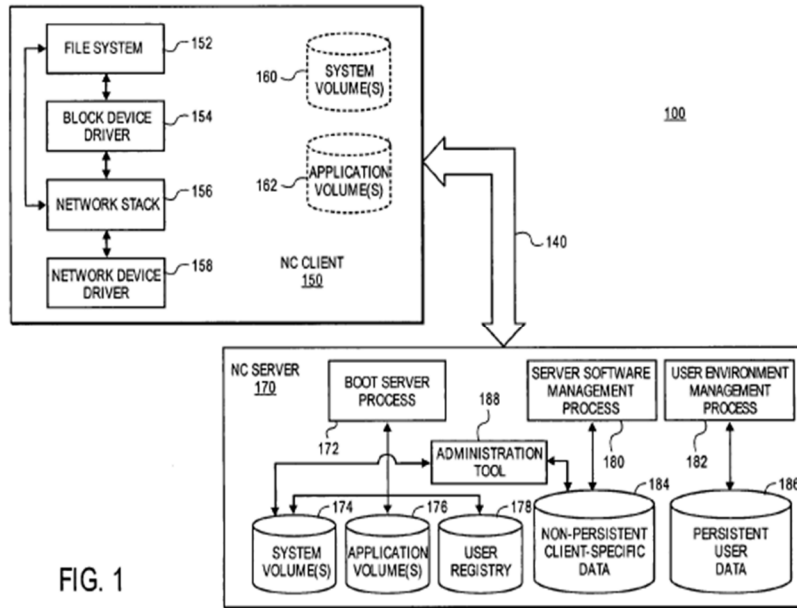
EX1005 (Birse), 12:30-52.

282. **In sum**, Birse discloses wherein the root image [Birse's system volumes 650, 660] is concurrently accessible to a plurality of compute nodes [Birse's NC clients 150].

5. Claim 6: The system as recited in claim 1 wherein the first storage unit is remotely located from the compute node.

283. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

284. As I explained regarding the independent claims, Birse's NC client 150 (compute node) is remote from the first storage unit (NC folders 630/640 of hard drive 610) because the first storage unit in Birse is on the NC server 170, which is remote from the NC client 150, as the NC server 170 and NC client 150 communicate over a data communication link 140, as shown in FIGS. 1 and 6 below.



285. Birse further specifies that the storage on the NC server (*e.g.*, hard drive 610) is a “remote storage medium on the NC server.” EX1005 (Birse), 10:56-62.

286. **In sum**, Birse discloses wherein the first storage unit [Birse’s NC admin/shared directories on hard drive 610] is remotely located from the compute node [Birse’s NC client 150].

6. Claim 7: The method as recited in claim 7 wherein the second storage unit is remotely located from the compute node.

287. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

288. As I explained regarding the independent claims, Birse’s NC client 150 (compute node) is remote from the second storage unit (NC folders 680/695 of hard drive 610) because the second storage unit in Birse is on the NC server 170, which is remote from the NC client 150, as the NC server 170 and NC client 150 communicate over a data communication link 140, as shown in FIGS. 1 and 6 below.

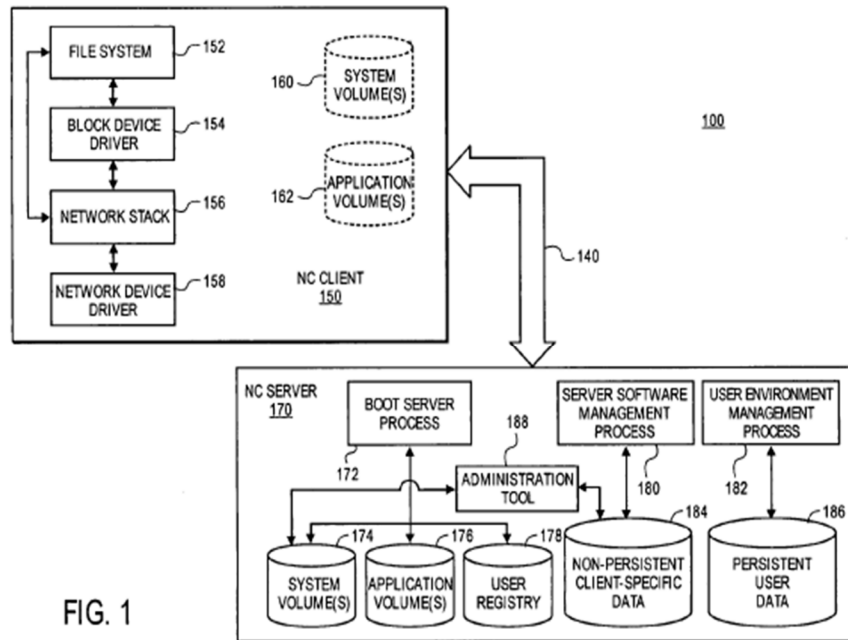


FIG. 1

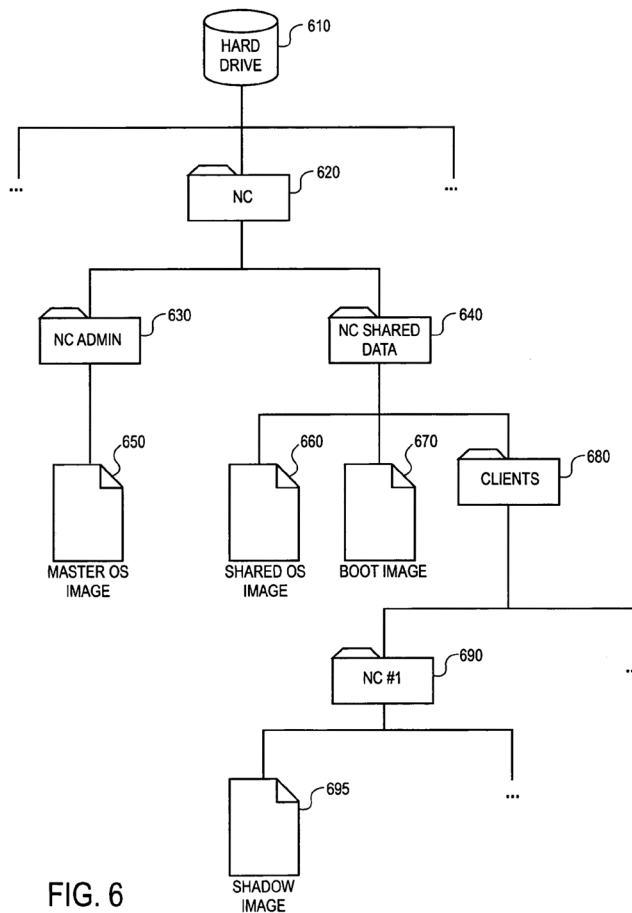


FIG. 6

289. Other portions of Birse emphasize the second storage unit is “remote” because Birse specifies that read/write requests to the shadow volumes on the second storage unit are redirected by the block device driver 154 “to a remote storage medium on the NC server.” EX1005 (Birse), 10:56-62.

290. **In sum**, Birse discloses wherein the second storage unit [Birse’s NC client directory 690 on hard disk 610] is remotely located from the compute node [Birse’s NC client 150].

7. Claim 8: The system as recited in claim 1 wherein the second storage unit contains a block modification log for the compute node.

291. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

292. As described in §§XII.A.1.d above and XII.A.12.d below with respect to claim elements [1.3] and [15.3], Birse discloses storing a leaf image (shadow volume/image 930, 695) for respective compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826) on a second storage unit (NC client folders 690 #1...n on hard drive 610), where the leaf image includes new data blocks of the root image (write requests for new data) and changes to the blocks of the root image (write request for changes to operating system data): “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.” EX1005 (Birse), 12:54–59.

293. Birse’s shadow volume “shadows the user system volume 924 by *storing modifications* that are made to the user system volume 924.” *Id.*, 14:49-53 (emphasis added). Therefore, the shadow volumes in Birse serve as block modification logs in that they track blocks that have been modified as compared to the original system volumes.

294. **In sum**, Birse discloses wherein the second storage unit [Birse’s NC client directory 690 on hard disk 610] contains a block modification log [Birse’s shadow image 695] for the compute node [Birse’s NC client 150].

8. Claim 9: The system as recited in claim 1 wherein the first storage unit in [sic] contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.

295. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

296. As an initial matter, the ’282 patent does not define or even use the term “partition” in its written description, only in the claims. Given that, and as I describe in §VII.B above, a POSITA would have understood that the claimed partitions here can be any logical storage area on a hard disk: a folder or directory, a raw partition, an area of a database, etc. To the extent Patent Owner argues that the claimed partitions here must be raw partitions of a physical hard disk, a POSITA would have

understood that a hard disk (*e.g.*, SCSI hard disk /dev/sda) can be easily partitioned using standard tools into several partitions (*e.g.*, /dev/sda1, /dev/sda2, /dev/sda3, and so on).

297. Birse discloses that the first storage unit (holding the core system volumes and the user system volume), and the second storage units (holding the shadow system volumes), are both contained on a same hard disk, but in different locations/partitions: “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.” EX1005 (Birse), 2:24–27.

298. “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.” EX1005 (Birse), 12:30–37. Birse’s Figure 6 is reproduced below:

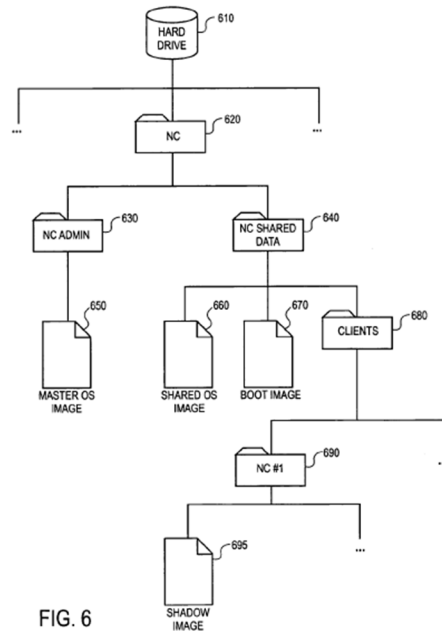


FIG. 6

299. **In sum**, Birse discloses wherein the first storage unit is contained within a first partition on a hard disk [Birse's NC admin/shared directories on hard drive 610] and the second storage unit is contained within a second partition on the hard disk [Birse's NC client directory 690 on hard disk 610].

9. Claim 10: The system as recited in claim 1 wherein the union block device writes sector X on the second storage unit.

300. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

301. For these elements, *see* claim limitations for Claim 15, element [15.7] in this Ground 1 (§XII.A.12.h below).

302. **In sum**, Birse discloses wherein the union block device writes sector X [Birse's blocks of shadow image 695] on the second storage unit [Birse's NC client directory 690 on hard disk 610].

10. Claim 13: The system as recited in claim 1 wherein the first storage unit is read only.

303. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

304. Birse further discloses that the first storage unit is read-only protected: “The system volumes 174 include a protected, read-only, master copy of the operating system software.” EX1005 (Birse), 5:13-14. Additionally, although Birse identifies that system volumes 660/924 accessed by the NC clients are “read-write” files (see FIG. 9 below), these system volumes are also effectively treated as “read only” in various embodiments of Birse.

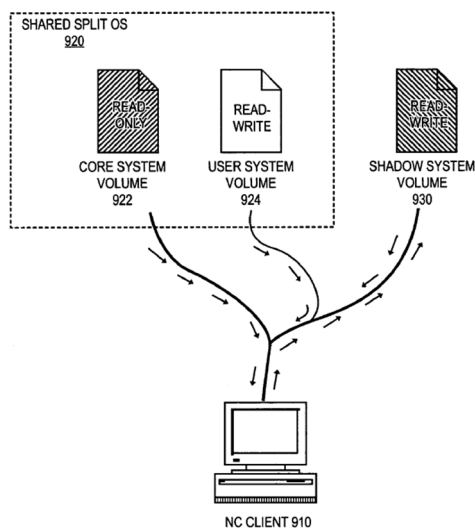


FIG. 9

305. As I explained regarding the independent claims, in Birse, write requests to a system volume (*i.e.*, “said first storage unit”) are redirected by the block

device driver by translation into “a write request directed to the user’s shadow volume on the NC server.” *See* §§XII.A.1.e-XII.A.1.h above; EX1005 (Birse), 11:8–11. Thus, because all write requests directed to a system volume are redirected to the user’s shadow volume, the system volume is “read-only” as required by claims 13 and 29.

306. **In sum**, Birse discloses wherein the first storage unit is read only [Birse’s system volumes 174, 650, 660, 922, 924 stored in NC admin/shared directories on hard drive 610].

11. Claim 14: The system as recited in claim 1 further comprising an intermediate image between the root image and the leaf image, wherein the intermediate image comprises an application environment service group.

307. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in this Ground 1.

308. As an initial matter, a POSITA would have understood that storage systems that can take snapshots, can take more than one snapshot, and hence the snapshots between the last and first one (or the base image) are in fact “intermediate” ones. *See* §VII.D above.

309. Birse renders claim 14 obvious. The ’282 Patent describes that an intermediate image comprises versions of the root image from which a compute node’s instance of the application environment is derived. EX1001 (’282 Patent), 4:58-61. The ’282 Patent also describes that, as shown in FIG. 2. below, a second

branch node 220 includes a service group 221 (service group 2.1) that “reflects a change to the application environment” and resides between the root image and the leaf images 150g-150i. EX1001 (’282 Patent), 5:41-52.

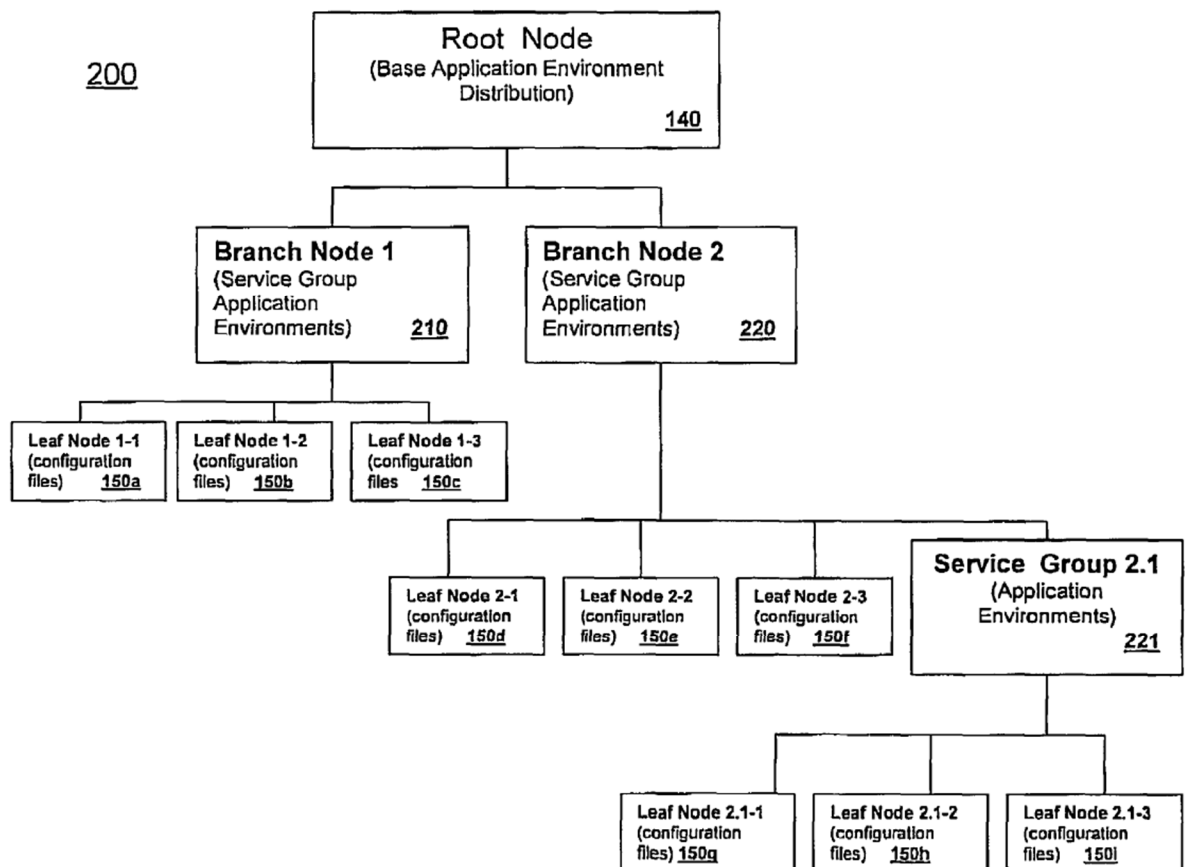


Figure 2

310. The ’282 Patent further explains that an application group (AG) “is a way to provide a group of compute nodes their own instantiation of an application environment” and that “[a]t creation time, an AG comprises the original root image.” EX1001 (’282 Patent), 5:19-29. Nodes shown in FIG. 2 can be “read-only” and others can be “read-write-modify.” EX1001 (’282 Patent), 5:32-33.

311. Likewise, as described herein with respect to independent claims 1 and 15, Birse discloses storing a master (read-only) OS image 650 and a shared OS image 660 (read-write), as seen in Birse's FIG. 6 below:

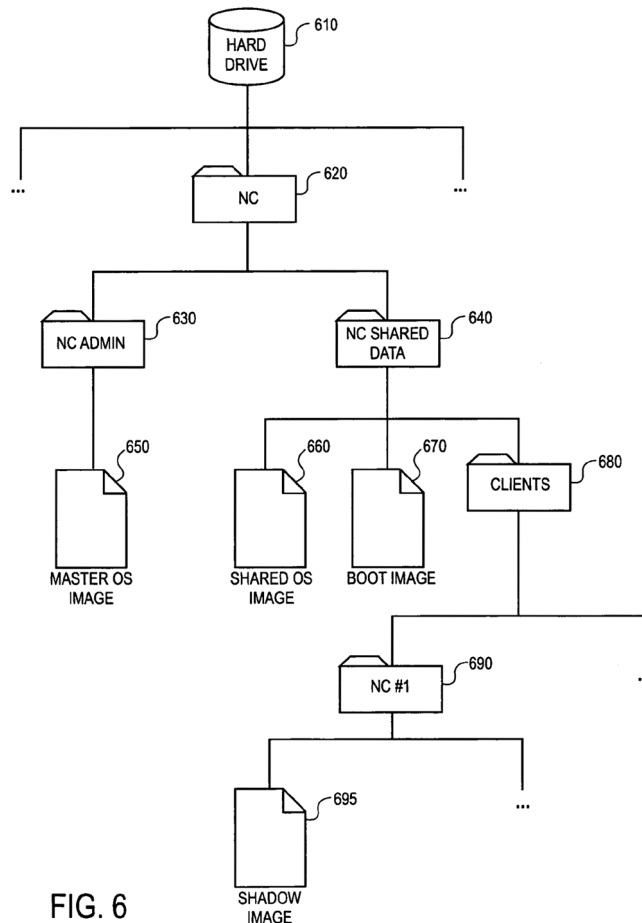


FIG. 6

312. In Birse, the shared OS image 660 can be stored in a third storage unit (NC shared data folder 640) and, while the “shared operating system image 660 is a read-write version of the read-only master operating system image 650” (EX1005 (Birse)), 12:44–47), changes can be made to the shared OS image 660 such that it acts as an intermediate image comprising an application environment service group.

313. Specifically, in Birse, an NC client 150 that “has appropriate access privileges” communicates with NC server 170 and NC server 170 copies one or more shared images, like the shared operating system image 650, to the NC client’s folder as a working copy. EX1005 (Birse), 13:9-22. Changes can then be made to the working copy and a new shared images (*e.g.*, a new shared OS image 660), can then be stored on the server 170 by the NC client committing the changes, which causes the new one or more shared images to become “available to any NC client that subsequently boots.” EX1005 (Birse), 13:20-14:8. This process is illustrated in FIG. 7 below.

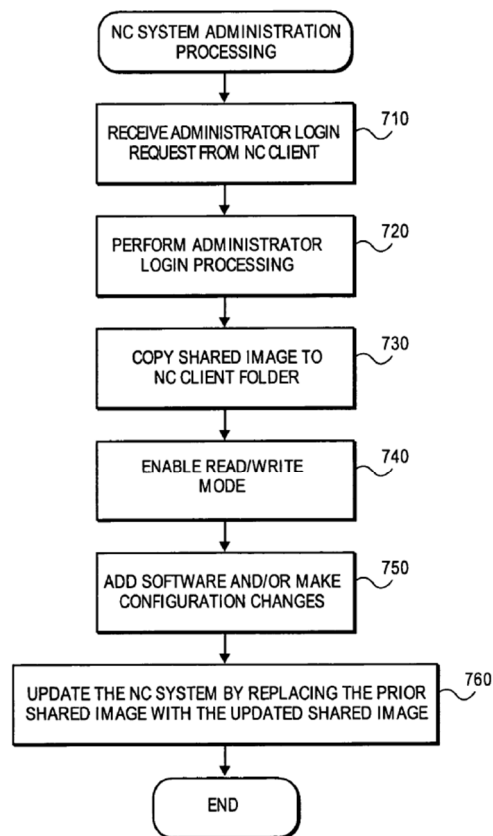


FIG. 7

314. Birse further explains that this intermediate image can be made available to a certain group of NC clients or distributed and made available to many other networked connected NC clients. See FIG. 8 of Birse, below.

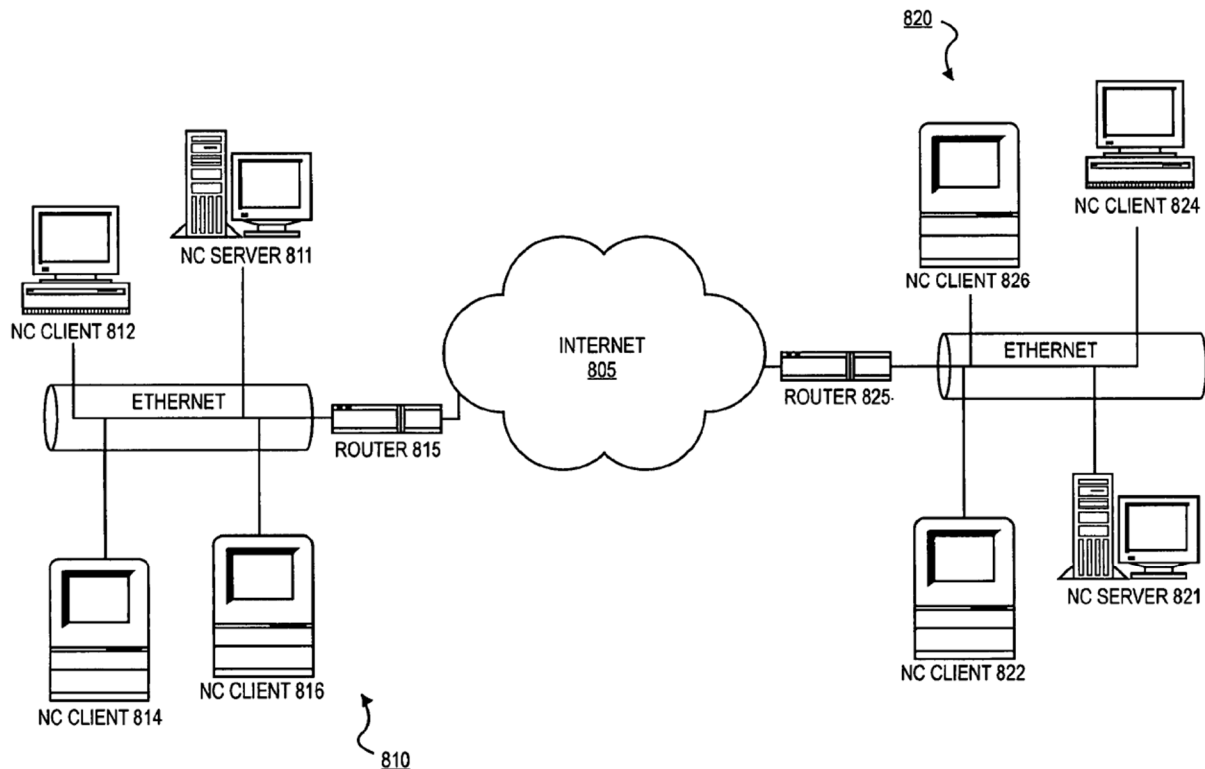


FIG. 8

315. Specifically, Birse describes, with reference to FIG. 8, that “two NC systems 810 and 820” can be “coupled via the Internet 805.” Birse contemplates “that the NC system configuration process described above with reference to FIG. 7 may also be useful in connection with the reconfiguration of remote NC systems” and that “[f]or example, the administration tool 188 may be run from any of NC clients 812, 814, or 816 to install a new application program on local NC system 810 that is desired by the users of the local NC system 810 as well as by the users of NC

clients 822, 824, and 826 of remote NC system 820.” Then, “[a]fter completing the steps described above for installation on the local NC system 810, the administration tool 188 running on NC system 810 may replicate the modified shared images to the remote NC system 820. EX1005 (Birser), 14:8-21.

316. Thus, as described above regarding this claim, in Birser, shared OS images 660 can be created that differ from the original master OS image (an intermediate image) on a third storage unit that is between the root image and the leaf image (as shown in FIG. 6 above where the changed shared OS image 660 is stored in folder 640 between the master OS image 650 and the shadow image(s) 695). This intermediate image in Birser comprises an application environment service group because, as explained above regarding this claim, in Birser, the intermediate image reflects a change from the original master OS image and is made available to NC clients so that this group of NC clients can use the revised application environment.

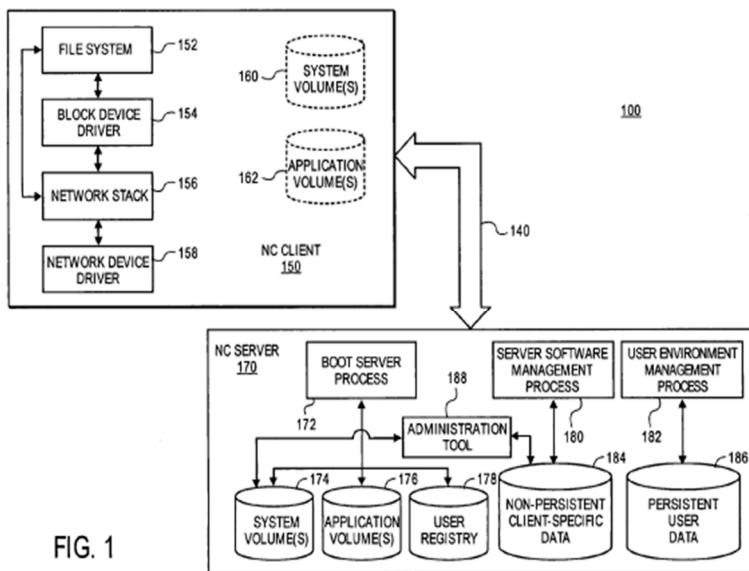
317. **In sum**, Birser discloses an intermediate image [Birser’s subsequent versions of system volumes 660, 924] between the root image [Birser’s system volumes 650, 922] and the leaf image [Birser’s shadow volumes 695 and 930], wherein the intermediate image comprises an application environment service group [Birser’s change from the original master OS image made available to a group of NC clients].

12. Claim 15

a. [15.0] A method for distributing an application environment comprising:

318. See also claim limitation [1.0] in this Ground 1 (§XII.A.1.a above).

319. To the extent the preamble is limiting, Birse discloses a method for distributing an application environment (from NC server 170) to a plurality of compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826): “[150] boot over the network by accessing a stored copy of the operating system from the NC server [170]. Additionally, when NC clients want to run an application, they access it from the NC server.” EX1005 (Birse), 4:56-60. See also *id.*, 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications). This is illustrated in Birse’s Figure 1, reproduced below:



320. FIG. 3 of Birse shows an example method of communication between an NC client 310 and an NC server 320:

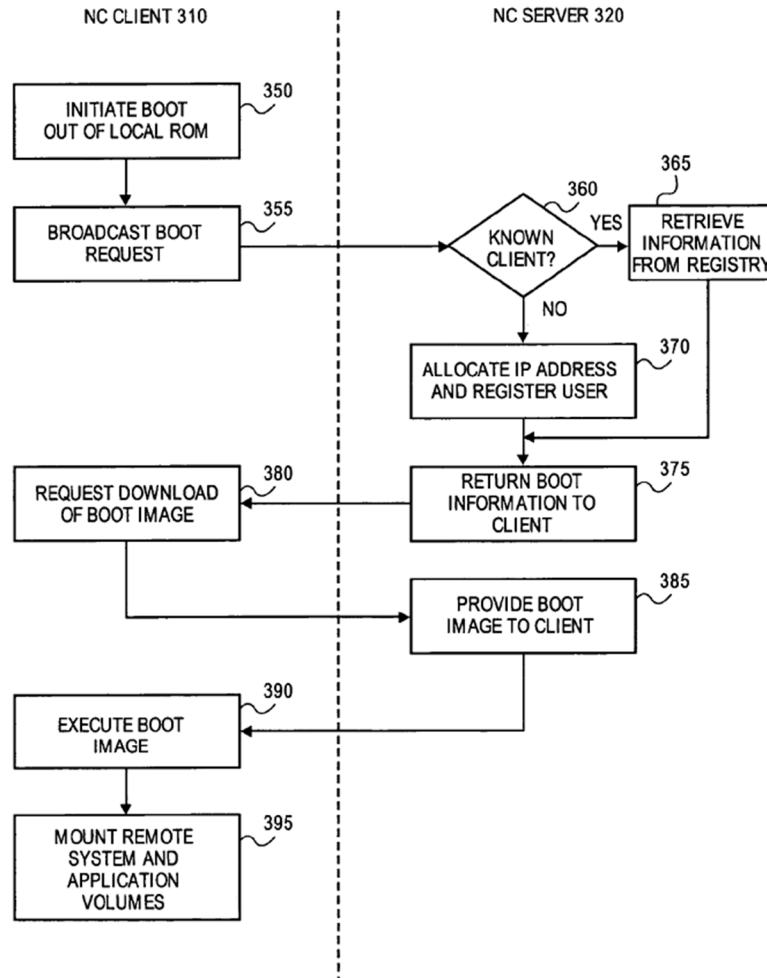


FIG. 3

321. **In sum**, Birse discloses a method for distributing an application environment.

- b. [15.1] storing blocks of a root image of a compute node on a first storage unit, wherein the compute node comprises a computer system,**

322. *See* claim limitation [1.1] in this Ground 1 (§XII.A.1.b above).

- c. [15.2] and wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;**

323. *See* claim limitation [1.2] in this Ground 1 (§XII.A.1.c above).

- d. [15.3] storing a leaf image comprising new data blocks and changes to the blocks of the root image on a second storage unit, wherein the second storage unit comprises a second non-volatile memory;**

324. *See* claim limitation [1.3] in this Ground 1 (§XII.A.1.d above).

- e. [15.4] merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit to create the application environment; and**

325. *See* claim limitation [1.5] in this Ground 1 (§XII.A.1.f above).

- f. [15.5] delivering the application environment to the compute node;**

326. *See* claim limitation [1.4] in this Ground 1 (§XII.A.1.e above).

- g. [15.6] wherein the modifying comprises: upon receiving a write request from the compute node for a sector X, creating an appropriate persistent mapping for sector X; and**

327. *See* claim limitation [1.7] in this Ground 1 (§XII.A.1.h above).

h. [15.7] writing sector X on the second storage unit.

328. *See* also claim limitation [1.7] in this Ground 1 (§XII.A.1.h above).

329. In Birse, writes to shadow volumes (and writes thus to sectors and/or blocks of the shadow volumes) are written on the second storage unit. As explained, the shadow volumes (and modifications/writes thereto) are on a second storage unit (e.g., NC client folders 690 #1...n on hard drive 610). EX1005 (Birse), 12:42-63.

330. As described in this Ground 1 for claim limitation [1.7] (§XII.A.1.h above), Birse discloses persistently mapping sectors for writing blocks of its shadow volumes/images, where blocks of the shadow volume (leaf image) that are modified in response to a write request (a sector X of the shadow volume) are persistently mapped (a “one-to-one block mapping in copy-on-write images”) and correspond to associated blocks of the original system volume (root image). EX1005 (Birse), 10:63-65, 11:6-17, 15:20-52.

331. **In sum**, Birse discloses writing sector X [Birse’s blocks of shadow volume] on the second storage unit [Birse’s NC client directory 690 on hard disk 610].

13. Claim 16: The method as recited in claim 15 further comprising: modifying the leaf image in response to the compute node's access to the application environment.

332. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

333. As I explained regarding claims 1 and 15, the '282 Patent explains that: UBDs 130*a-n* may also modify the leaf image *in response to their respective compute node's access to its instance of the application environment*. For example, upon receiving a write request from their respective compute nodes for a sector X, the UBDs 130*a-n* will create an appropriate persistent mapping for sector X and then write sector X onto their respective second storage units 150*a-n*, where sector X can then be modified.

EX1001 ('282 Patent), 5:1-11 (emphasis added).

334. Another portion of the '282 Patent states similar subject matter:

Step 350 is modifying the leaf image in response to the compute node's access to its instance of the application environment. The modifying of the leaf image may include copying one or more data blocks from the root image to the leaf image, and modifying the data block in the leaf image.

EX1001 ('282 Patent), 6:23-27.

335. As I also explained regarding claim 15, Birse discloses or suggests this subject matter. In Birse, "one or more shadow volumes [695], may be created for

the NC client 150.” 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.” EX1005 (Birse), 7:35-53; *see also id.*, 11:30-32.

336. Birse further explains that “the file system 152 generates a write request directed to the system volumes 160,” then 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, and finally “the NC server 170 stores information associated with the write request in the shadow volume associated with the NC client 150.” *Id.*, 10:63-65, 11:6-17. As shown for example in FIG. 10 of Birse below, blocks of the shadow volume (leaf image) that are modified in response to a write request (a sector X of the shadow volume) are persistently mapped (a “one-to-one block mapping in copy-on-write images”) and correspond to associated blocks of the original system volume (root image).

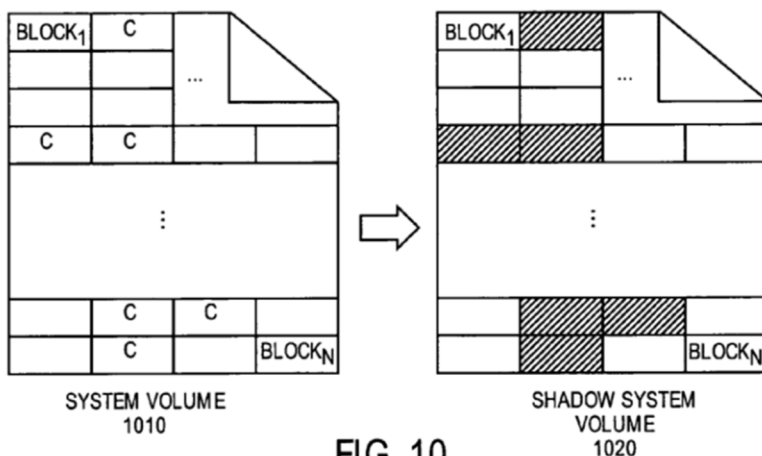


FIG. 10

Id., 10:63-65, 11:6-17, 15:33-34.

337. Birse further explains that NC clients “boot by accessing a copy of the operating system over the network” and “[w]hen 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.” EX1005 (Birse), 4:31-35, 9:2-5.

338. **In sum**, Birse discloses modifying the leaf image [Birse’s shadow volume 695, 924] in response to the compute node’s [Birse’s NC client 150] access to the application environment.

14. Claim 19: The method as recited in claim 15 wherein the compute node comprises a thin-client workstation.

339. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

340. For these elements, *see* claim limitations for Claim 3 in this Ground 1 (§XII.A.2 above).

15. Claim 20: The method as recited in claim 15 wherein the root image comprises an operating system.

341. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

342. For these elements, *see* claim limitations for Claim 4 in this Ground 1 (§XII.A.3 above).

16. Claim 21: The method as recited in claim 15 wherein the root image is concurrently accessible to a plurality of compute nodes.

343. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

344. For these elements, *see* claim limitations for Claim 5 in this Ground 1 (§XII.A.4 above).

17. Claim 22: The method as recited in claim 15 wherein the first storage unit is remotely located from the compute node.

345. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

346. For these elements, *see* claim limitations for Claim 6 in this Ground 1 (§XII.A.5 above).

18. Claim 23: The method as recited in claim 15 wherein the second storage unit is remotely located from the compute node.

347. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

348. For these elements, *see* claim limitations for Claim 7 in this Ground 1 (§XII.A.6 above).

19. Claim 24: The method as recited in claim 15 wherein the second storage unit contains a block modification log for the compute node.

349. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

350. For these elements, *see* claim limitations for Claim 8 in this Ground 1 (§XII.A.7 above).

20. Claim 25: The method as recited in claim 15 wherein the first storage unit is contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.

351. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

352. For these elements, *see* claim limitations for Claim 9 in this Ground 1 (§XII.A.8 above).

21. Claim 26: The method as recited in claim 15 wherein merging occurs at an operational level between the first and second storage units and file system of the compute node.

353. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

354. For these elements, *see* claim limitations [1.4]-[1.6] (§§XII.A.1.e-XII.A.1.g above) in this Ground 1 pertaining to the union block device. Birse discloses these elements for the same reasons.

22. Claim 29: The method as recited in claim 15 wherein the first storage unit is read only.

355. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

356. For these elements, *see* claim limitations for Claim 13 in this Ground 1 (§XII.A.10 above).

23. Claim 30: The method as recited in claim 15 further comprising: creating an intermediate image on a third storage unit between the root image and the leaf image, wherein the intermediate image comprises an application environment service group.

357. For the method as recited in claim 15, *see* claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) and [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in this Ground 1.

358. For these elements, *see* claim limitations for Claim 14 in this Ground 1 (§XII.A.11 above).

B. Ground 2: The Combination of Birse and Ballew Renders Claims 2, 11, 12, 18, 27, and 28 Obvious.

1. Motivations to Combine Birse and Ballew

359. Both Birse and Ballew are concerned with management of computing devices over a network and booting and/or initializing the computing devices with an operating system. For example, as explained in §§X.A above and §§XII.A.1.a-XII.A.1.h above, Birse discloses computing devices that boot using a net-booted environment “by accessing a copy of the operating system over a network,” and in which one or more shadow volumes are created from the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing changes to the operating system made during execution by a particular computing device. EX1005 (Birse), 4:31-35, 7:41-46.

360. As explained in §X.B above, Ballew is concerned with an HPC system that uses HPC clusters of a server 102 with a grid 110 that “is a group of nodes 115 interconnected for increased processing power” for dynamically allocating processing of tasks. EX1006 (Ballew), [0016]-[0017]. Like Birse, Ballew discusses that when an HPC server 102 “is first initialized or booted” “a unique execution environment” is provided “for each allocated node of virtual cluster 220,” where the

execution environment can include an “operating system” and “a set of installed applications and data.” EX1006 (Ballew), [0029].

361. A POSITA would have been motivated to modify Birse and/or Ballew such that the net-booted operating system processes disclosed in Birse could be used with HPC servers and associated nodes like those disclosed in Ballew. At least one reason is that, while Ballew discloses providing an execution environment generally to nodes in an HPC system, Birse offers details on OS storage and distribution (*e.g.*, Birse’s master OS images and shadow images), such that a master OS image is stored by the server 102 in Ballew and the operating system of the execution environment for each node of the server 102 can be distributed using Birse’s shadow volumes.

362. Modifying Birse with Ballew’s HPC servers/cluster teachings would have been simple to a POSITA and would not have required extensive experimentation or imposed undue burden – especially considering both contemplate distribution of application environments to computing devices. For the same reasons, the combination would have yielded predictable results (as HPC servers were well-understood to improve task processing power and allocation). A POSITA would also have a high expectation of success, as Ballew describes a need for distributing execution environments to nodes, and Birse provides a mechanism to do so.

2. Claim 2: The system as recited in claim 1 wherein the compute node comprises a server.

363. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in Ground 1.

364. Ballew discloses, as part of a server 102, a node that can “comprise any computer or processing device such as, for example, blades, general-purpose personal computers (PC), Macintoshes, workstations, Unix-based computers, or any other suitable devices.” EX1006 (Ballew), [0016]-[0017].

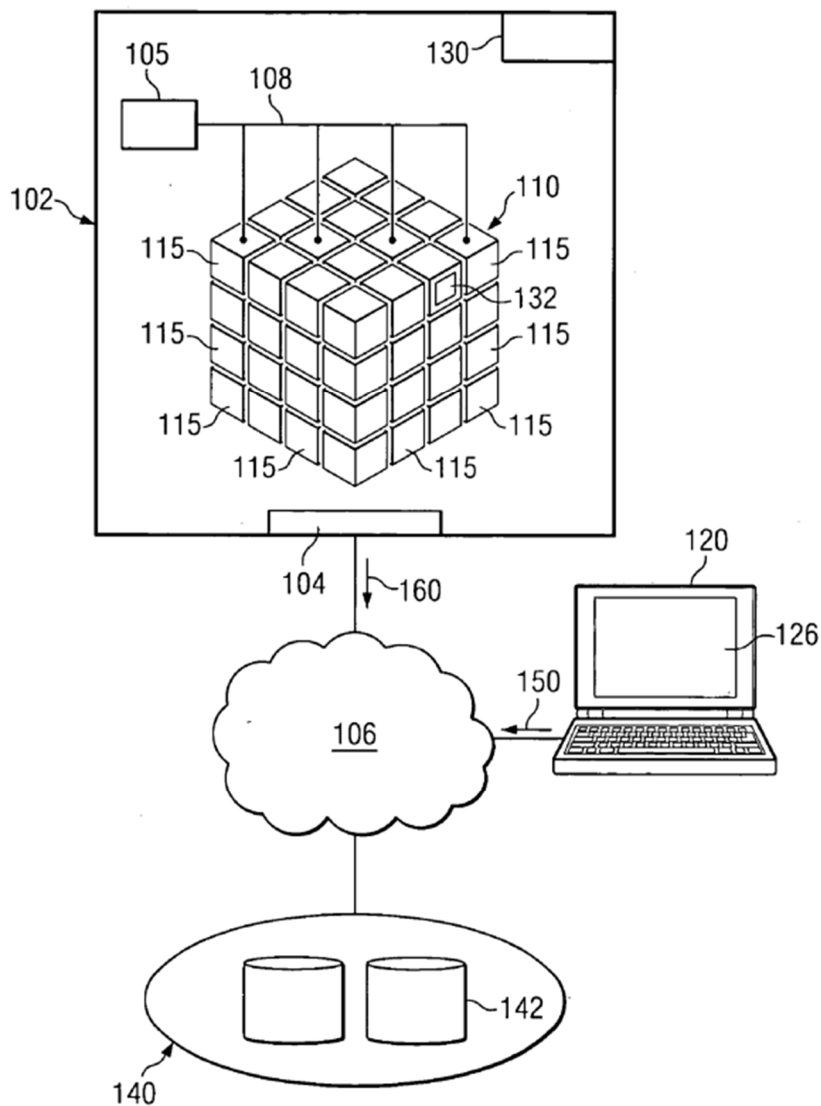
365. As explained in §§XII.A.1.a-XII.A.1.h above and §§XII.A.12.a-XII.A.12.h above, Birse meets all elements of independent Claims 1 and 15. Furthermore, as explained in the motivations to combine Birse and Ballew (§XII.B.1 above), it would have been obvious to use Birse’s OS distribution processes with a server as described in Ballew, as Ballew discusses a need to distribute an operating system as part of an execution environment to Ballew’s nodes.

366. **In sum**, the combination of Birse and Ballew discloses wherein the compute node [Birse’s NC client 150] comprises a server [Ballew’s server 102].

3. Claim 11: The system as recited in claim 1 wherein the system operates in a high performance computing cluster.

367. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in Ground 1.

368. As explained in §§XII.A.1.a-XII.A.1.h above and §§XII.A.12.a-XII.A.12.h above, Birse meets all elements of independent Claims 1 and 15.



370. In Ballew, when the HPC server 102 “is first initialized or booted” a “cluster management engine 130 may dynamically allocate various portions of grid 110 to one or more virtual clusters 220 based on, for example, predetermined policies,” and can “also manage and provide a unique execution environment for each allocated node of virtual cluster 220.” EX1006 (Ballew), [0029]. Ballew explains that “[t]he execution environment may consist of the hostname, IP address, operating system, configured services, local and shared file systems, and a set of installed applications and data.” *Id.*

371. Furthermore, as explained in the motivations to combine Birse and Ballew (§XII.B.1 above), high performance computing clusters were well-known prior to the '282 Patent, and it would be obvious to operate systems and methods that deal with distributing application environments, like that of Birse, in a high performance computing cluster. The Background of the '282 Patent itself indicates that HPC clusters were known prior to the '282 Patent, explaining that “[o]ver the years, as the interne [sic] has expanded and computers have multiplied, the need for clustered computing such as High Performance Computing (HPC) has increased” and that “[c]lustered computing involves multiple compute nodes, usually a server grid, that work together to achieve a common task.” EX1001 ('282 Patent), 1:20-25. The '282 Patent further asserts that there were issues with “low cluster bring-up time” because complete clones of master OS images would have to be created for

each node in the cluster and those complete clones would have to be updated to update the OS of the cluster. *Id.*, 1:39-57.

372. Thus, Birse as combined with Ballew provides a solution to this same problem, such that one master OS image can be stored and the operating system of the execution environment for each node of Ballew's server 102 can be distributed using Birse's shadow volumes.

373. **In sum**, the combination of Birse and Ballew discloses wherein the system operates in a high performance computing cluster [Ballew's HPC system 100].

4. Claim 12: The system as recited in claim 1 wherein the system operates in a grid computing cluster.

374. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in Ground 1.

375. As explained above regarding Claims 11 (§XXX) and 27 (§XXX), the Background of the '282 Patent admits that clustered computing "involves multiple compute nodes, usually a server grid." EX1001 ('282 Patent), 1:20-25.

376. Similarly, the clustered computing described in Ballew uses a grid 110 comprising the plurality of nodes 115, where the grid 110 "is a group of nodes 115 interconnected for increased processing power." EX1006 (Ballew), [0016]-[0017]. Ballew's FIGS. 2C and 2D, shown below, illustrate examples in which a grid 210 has virtual clusters 220 allocated within the grid.

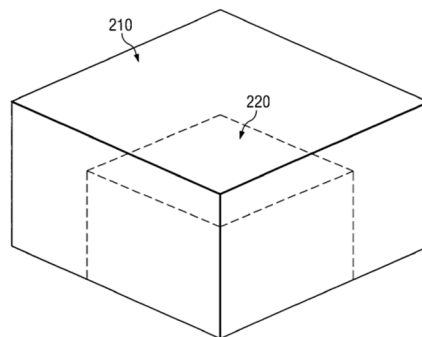


FIG. 2C

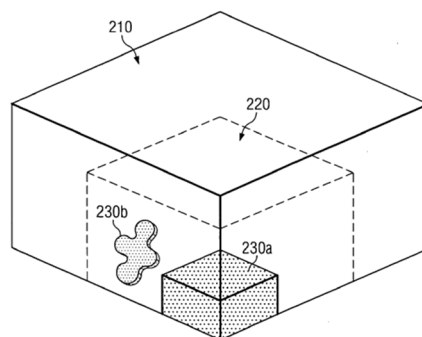


FIG. 2D

377. Ballew indicates that multiple servers 102, *e.g.*, a “server pool,” each having a grid can also be used. EX1006 (Ballew), [0016]-[0017].

378. **In sum**, the combination of Birse and Ballew discloses wherein the system operates in a grid computing cluster [Ballew’s grid 110 of HPC system 100].

5. Claim 18: The method as recited in claim 15 wherein the compute node comprises a server.

379. For the method as recited in claim 15, *see* claim limitations [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in Ground 1.

380. For these elements, *see* claim limitations for Claim 2 in this Ground 2 (§XII.B.2 above).

6. Claim 27: The method as recited in claim 15 wherein the method operates in a high performance computing cluster.

381. For the method as recited in claim 15, *see* claim limitations [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in Ground 1.

382. For these elements, *see* claim limitations for Claim 11 in this Ground 2 (§XII.B.3 above).

7. Claim 28: The method as recited in claim 15 wherein the system operates in a grid computing cluster.

383. For the method as recited in claim 15, *see* claim limitations [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in Ground 1.

384. For these elements, *see* claim limitations for Claim 12 in this Ground 2 (§XII.B.4 above).

C. Ground 3: The Combination of Birse and Wu Renders Claims 8 and 24 Obvious.

1. Motivations to Combine Birse and Wu

385. Both Birse and Wu are concerned with “copy-on-write” schemes in which reads or writes directed to a main volume are redirected to a virtual volume. *Compare* EX1005 (Birse), 14:56-61 (“copy-on-write” redirecting to a shadow volume) *with* EX1007 (Wu), 4:61-5:4 (“copy-on-write” redirecting to a snapshot). For example, as explained in §§X.A above and XII.A above, Birse discloses creating “one or more shadow volumes” “for the purpose of storing changes to the operating

system [stored in a core system volume] made during execution by a particular computing device.” EX1005 (Birse), 7:41-46.

386. As I explained in §X.C above, Wu discloses using a modification log to track modifications to the original volume between any two snapshots, such that, “[e]ach time a modification is detected, modification manager 212 may log the time at which the modification occurred, the address(es) of the modified block(s), and the pre- and post-modification values of the modified block(s).” EX1007 (Wu), 6:14-23. In the event a snapshot is deleted, the modifications stored in the modification log can be used to reconstruct the snapshot using another snapshot. EX1007 (Wu), 7:26-29. Wu explains that storing snapshot volumes can be storage intensive and thus proposes using its smaller footprint modification logs to track block changes. EX1007 (Wu), 1:66-2:42. *See also* §VII.D above.

387. A POSITA would have been motivated to modify Birse based on Wu’s disclosures to include Wu’s modification log in Birse’s system such that block modifications to Birse’s system volumes are tracked in addition to, or in place of, Birse’s shadow volumes. Doing so would have provided the Birse system with another way to track modifications to system volumes with a more efficient disk space usage. Furthermore, it would have additionally allowed for deleted shadow volumes to be restored if needed using Wu’s modification log.

388. Modifying Birse to use Wu’s modification log teachings would have been simple to a POSITA and would not have required extensive experimentation or imposed undue burden – especially considering that both contemplate management of copy-on-write volumes. For the same reasons, the combination would have yielded predictable results, and a POSITA would also have a high expectation of success, as both Birse and Wu describe a need for limiting disk space of shadow volumes and snapshots and Wu’s modification log provides a detailed solution to this problem while also providing a mechanism for restoring deleted data. EX1005 (Birse), 15:15-42; EX1007 (Wu), 1:66-2:42.

2. Claim 8: The system as recited in claim 1 wherein the second storage unit contains a block modification log for the compute node.

389. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in Ground 1.

390. To the extent Birse does not explicitly disclose a block modification log, Wu discloses a block modification log that is used to log and track modifications to an original volume between any two snapshots. EX1007 (Wu), 6:14-20. As shown in Wu’s FIG. 2, below, the block modification log includes “the time at which the modification occurred, the address(es) of the modified block(s), and the pre- and post-modification values of the modified block(s).” Birse combined with Wu, as described in the motivation-to-combine §XII.C.1 above, would provide for a

modification log to be used in Birse's system to track changes to system volume blocks.

Modification Log 260

	Timestamp	Block Address	Pre-Modification Value	Post-Modification Value
Modification 1	Between T1 & T2	Address 1	Value A	Value B
Modification 2	Between T1 & T2	Address 2	Value X	Value Y
Modification 3	Between T2 & T3	Address 3	Value C	Value D

FIG. 5

391. **In sum**, the combination of Birse and Wu discloses wherein the second storage unit [Birse's NC client directory 690 on hard disk 610] contains a block modification log [Wu's block modification log 260] for the compute node [Birse's NC client 150].

3. Claim 24: The method as recited in claim 15 wherein the second storage unit contains a block modification log for the compute node.

392. For the method as recited in claim 15, *see* claim limitations [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in Ground 1.

393. For these elements, *see* claim limitations for Claim 8 in this Ground 3 (§XII.C.2 above).

D. Ground 4: The Combination of Birse and Kleinschnitz Renders Claims 9, 17, and 25 Obvious.

1. Motivations to Combine Birse and Kleinschnitz

394. Both Birse and Kleinschnitz are concerned with storing changes to system volumes for use in virtual environments. *Compare* EX1005 (Birse), 14:26-15:7 (using shadow volumes to store changes to system volume) *with* EX1008 (Kleinschnitz), [0003], [0007], [0016], [0044] (using snapshots to store incremental changes to a compute state).

395. For example, as explained in §§X.A above and XII.A above, Birse discloses computing devices that boot using a net-booted environment “by accessing a copy of the operating system over a network,” and in which one or more shadow volumes are created from the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing changes to the operating system made during execution by a particular computing device. EX1005 (Birse), 4:31-35, 7:41-46.

396. As explained in §X.D above, Kleinschnitz discloses using snapshots that include “incremental images” reflecting “the state of the computer’s storage” and storing the snapshots at another storage location. EX1008 (Kleinschnitz), [0003], [0035], [0059]. Kleinschnitz explains that “an incremental image 104 may be merged with a prior complete snapshot image 104 to form a complete and more recent snapshot image.” EX1008 (Kleinschnitz), [0071]. Kleinschnitz also

discloses that storage disks can be partitioned, such as “by placing user data in one partition and system data and software in another partition.” EX1008 (Kleinschnitz), [0079].

397. A POSITA would have been motivated to modify Birse based on Kleinschnitz’s disclosure to include Kleinschnitz’s merging/reconciliation of incremental images in Birse’s system such that the one or more shadow volumes are merged to create an updated complete volume. A POSITA would have understood that this “reconciling” is nothing more than merging snapshots, a process that naturally happens when snapshots are removed or “collapsed” together to make room. *See* §VII.D above.

398. A POSITA would have also been motivated to modify Birse based on Kleinschnitz’s disclosure to store system volumes (650, 660, 922, and 924) and shadow volumes (695 and 930) in different partitions so that, like in Kleinschnitz, user data (shadow volumes) are stored in one partition and system data (system volumes) are stored in another partition.

399. Modifying Birse to use Kleinschnitz’s merging/reconciliation and partition would have been simple to a POSITA and would not have required extensive experimentation or imposed undue burden – especially considering both contemplate management and storage of system volume changes. For the same reasons, the combination would have yielded predictable results and a POSITA

would also have a high expectation of success, as both Birse and Kleinschnitz describe snapshot reconciliation and disk partitioning techniques that were both already well-known before the '282 Patent.

2. Claim 9: The system as recited in claim 1 wherein the first storage unit is contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.

400. For the system as recited in claim 1, see claim limitations [1.0]-[1.7] (§§XII.A.1.a-XII.A.1.h above) in Ground 1.

401. Kleinschnitz discloses that storage disks can be partitioned, such as “by placing user data in one partition and system data and software in another partition.” EX1008 (Kleinschnitz), [0079]. As explained in §XII.D.1 above, it would have been obvious to modify Birse based on Kleinschnitz’s disclosure to store Birse’s system volumes (650, 660, 922, and 924) and shadow volumes (695 and 930), shown, *e.g.*, in FIGS. 6 and 9 of Birse below, in different partitions such that, like in Kleinschnitz, user data (shadow volumes) are stored in one partition and system data (system volumes) are stored in another partition.

402. Partitioning the data in this way would have served to create a clear distinction between storage/access locations for system volumes and shadow volumes.

403. **In sum**, the combination of Birse and Kleinschnitz discloses wherein the first storage unit [Birse’s NC/admin directories] is contained within a first

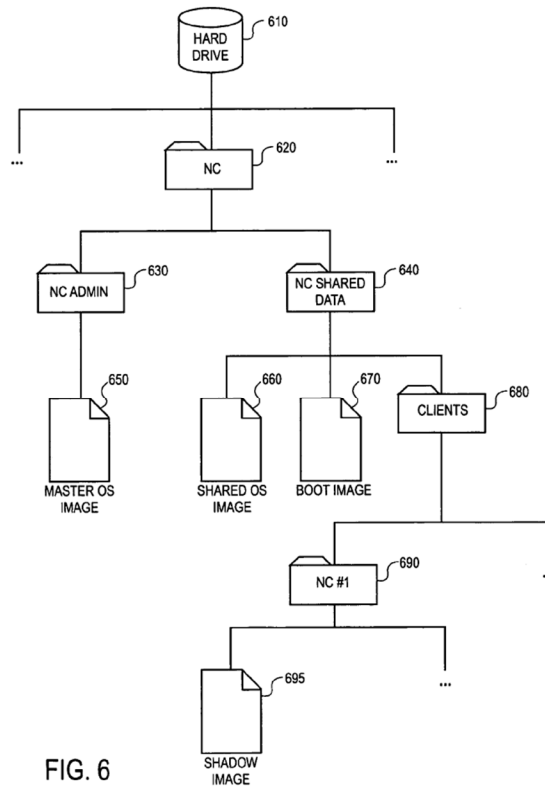
partition [Kleinschnitz's first partition] on a hard disk [Birse's hard disk 610] and the second storage unit [Birse's NC client directory 680] is contained within a second partition [Kleinschnitz's second partition] on the hard disk [Birse's hard disk 610].

3. Claim 17: The method as recited in claim 15 further comprising: reconciling the root image and the leaf image to form a new root image.

404. For the method as recited in claim 15, *see* claim limitations [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in Ground 1.

405. A POSITA would have understood that this “reconciling” is nothing more than merging snapshots, a process that naturally happens when snapshots are removed or “collapsed” together to make room. *See* §VII.D above.

406. Kleinschnitz discloses using snapshots that include “incremental images” reflecting “the state of the computer’s storage” and storing the snapshots at another storage location. EX1008 (Kleinschnitz), [0003], [0035], [0059]. Kleinschnitz explains that “an incremental image 104 may be merged with a prior complete snapshot image 104 to form a complete and more recent snapshot image.” EX1008 (Kleinschnitz), [0071].



407. As explained in §XII.D.1 above, a POSITA would have been motivated to modify Birse based on Kleinschnitz’s disclosure to include Kleinschnitz’s merging/reconciliation of incremental images in Birse’s system such that the one or more shadow volumes are merged to create an updated complete volume. A POSITA would have understood that this “reconciling” is useful to make room in storage for new shadow images. The ’282 Patent provides the same reasons for reconciling its leaf images with the root images, stating such reconciliation “may be desirable if, for example, the leaf image has grown to exceed a particular disk quota.” EX1001 (’282 Patent), 6:30-32.

408. **In sum**, the combination of Birse and Kleinschnitz discloses reconciling [Kleinschnitz's merging incremental images 104] the root image [Birse's system volumes 650, 660, 922, 924] and the leaf image [Birse's shadow volumes 695 and 930] to form a new root image [Kleinschnitz's updated/merged complete volume].

4. Claim 25: The method as recited in claim 15 wherein the first storage unit is contained within a first partition on a hard disk and the second storage unit is contained within a second partition on the hard disk.

409. For the method as recited in claim 15, *see* claim limitations [15.0]-[15.7] (§§XII.A.12.a-XII.A.12.h above) in Ground 1.

410. For these elements, *see* claim limitations for Claim 9 in this Ground 4 (§XII.D.2 above).

XIII. SECONDARY CONSIDERATIONS

411. I have seen no evidence in either the '282 Patent's prosecution history or elsewhere supporting any secondary considerations arguments, or evidence of nexus of such alleged evidence to the challenged claims. To the extent that patent owner or its expert asserts the existence of any secondary considerations in its responses, I reserve the right to address any such evidence.

XIV. CONCLUSION

412. The findings and opinions set forth in this declaration are based on my work and examinations to date.

413. I may continue my examinations. I may also receive additional documentation and other factual evidence over the course of this IPR that will allow me to supplement and/or refine my opinions. I reserve the right to add to, alter, or delete my opinions and my declaration upon discovery of any additional information. I reserve the right to make such changes as may be deemed necessary.

414. In signing this declaration, I recognize that the declaration will be filed as evidence in an IPR before the PTAB. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

415. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under §1001 of Title 18 of the United States Code.

Date: October 10, 2025

A handwritten signature in black ink, appearing to read "Erez Zadok". The signature is fluid and cursive, with the first name "Erez" and last name "Zadok" clearly distinguishable.

Erez Zadok, Ph.D.