

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re <i>Ex Parte</i> Reexamination of:	)	
	)	
U.S. Patent No. 8,370,416	)	
	)	
Issued: February 5, 2013,	)	Group Art Unit: To Be Assigned
	)	
Named Inventor: David J. Hoover, et al.	)	Examiner: To Be Assigned
	)	
Control Number: To Be Assigned	)	
	)	
Filed: April 26, 2006	)	
	)	
Title: COMPATIBILITY ENFORCEMENT IN	)	
CLUSTERED COMPUTING SYSTEMS	)	
	)	
	)	
	)	

Mail Stop *Ex Parte* Reexam

Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

REQUEST FOR *EX PARTE* REEXAMINATION

Requester NetApp, Inc., respectfully requests *ex parte* reexamination under 35 U.S.C. § 302 and 37 C.F.R. § 1.510 of U.S. Patent No. 8,370,416 (“the ’416 patent,” Ex. A), assigned to Patent Owner Valtrus Innovations Limited, per the assignment at reel/frame: 055403/0001. Prior art not cited or applied during prosecution renders issued claims 1-18 unpatentable under 35 U.S.C. §§ 102 and 103.

As this Request demonstrates, prior art references, including at least *Baratti* and *Wyatt*, raise substantial new questions of patentability (“SNQs”) with respect to claims 1-18. Indeed, this prior art renders these claims unpatentable under 35 U.S.C. §§ 102 and 103. NetApp thus

respectfully requests that the Office grant this Request for *ex parte* reexamination of the '416 patent and cancel claims 1-18.

In accordance with 37 C.F.R. § 1.510(b), this Request includes the following:

- (1) A statement pointing out each substantial new question of patentability based on prior patents and printed publications.
- (2) An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited prior art to every claim for which reexamination is requested. For each statement of the patent owner and accompanying information submitted pursuant to [37 C.F.R.] § 1.501(a)(2) which is relied upon in the detailed explanation, the request must explain how that statement is being used to determine the proper meaning of a patent claim in connection with the prior art applied to that claim and how each relevant claim is being interpreted. If appropriate, the party requesting reexamination may also point out how claims distinguish over cited prior art.
- (3) A copy of every patent or printed publication relied upon or referred to in paragraph (b) (1) and (2) of this section accompanied by an English language translation of all the necessary and pertinent parts of any non-English language patent or printed publication.
- (4) A copy of the entire patent including the front face, drawings, and specification/claims (in double column format) for which reexamination is requested, and a copy of any disclaimer, certificate of correction, or reexamination certificate issued in the patent. All copies must have each page plainly written on only one side of a sheet of paper.
- (5) A certification that a copy of the request filed by a person other than the patent owner has been served in its entirety on the patent owner at the address as provided for in [37 C.F.R.] § 1.33(c). The name and address of the party served must be indicated. If service was not possible, a duplicate copy must be supplied to the Office.
- (6) A certification by the third party requester that the statutory estoppel provisions of 35 U.S.C. [§] 315(e)(1) or 35 U.S.C. [§] 325(e)(1) do not prohibit the requester from filing the *ex parte* reexamination request.

Attachments

- (1) Certificate of Service on Patent Owner.
- (2) Information Disclosure Statement Form PTO/SB/08.

Exhibits

- Ex. A: U.S. Patent No. 8,370,416 to David J. Hoover, et al. (“the ’416 patent”).
Ex. B: Patent Prosecution History of U.S. Patent No. 8,370,416.
Ex. C: Declaration of Professor Jon Weissman, PhD.
Ex. D: Curriculum Vitae of Jon Weissman, Ph.D.

Prior Art Exhibits

- PA-1: U.S. Patent No. 6,574,612 to Baratti et al. (“*Baratti*”)
PA-2: U.S. Patent Application Publication No. 2006/0036894 A1 to Bauer (“*Bauer*”).
PA-3: U.S. Patent No. 5,260,999 to Wyman (“*Wyman*”).
PA-4: U.S. Patent Application Publication No. 2001/0044782 to Hughes et al., (“*Hughes*”).
PA-5: RS/6000 43P Model 140 Product Description (“*Workstation Description*”).

Litigation Document Exhibits

- L-1: Plaintiff’s Complaint for Patent Infringement in *Valtrus Innovations Ltd. et al. v. NetApp, Inc.*, No. 2:25-cv-00517, Dkt. 1 (E.D. Tex. May 9, 2025)
L-2: Consolidation Order in *Valtrus Innovations Ltd. et al. v. NetApp, Inc.*, No. 2:25-cv-00517, Dkt. 11 (E.D. Tex. July 27, 2025)
L-3: *NetApp, Inc. v. Valtrus Innovations Ltd. et al.*, 1:25-cv-06138 (S.D.N.Y. July 25, 2025).

TABLE OF CONTENTS

	<u>Page</u>
I. RELATED PROCEEDINGS.....	1
A. Certification Regarding Statutory Estoppel.....	1
B. Other Proceedings.....	1
C. Request for Expedited Reexamination.....	1
II. IDENTIFICATION OF CLAIMS THAT SHOULD BE REEXAMINED.....	2
III. BACKGROUND OF THE TECHNOLOGY	2
A. It was known before April 2006 that clustered networks provided the advantages of redundancy and flexible scalability.	2
B. But it was also known that the flexibility inherent in clustered networks could wreak havoc on licensing management unless the system accounted for both node and cluster licensing information.	3
C. And it was known that the complexity of clustered networks required mechanisms to ensure nodes were compatible with each other and with the cluster.	4
IV. OVERVIEW OF THE '416 PATENT AND ITS PROSECUTION HISTORY	5
A. Specification	5
B. Claims 1-18 of the '416 Patent	7
C. Prosecution History.....	11
D. Level of Ordinary Skill in the Art.....	12
E. Claim Construction for This Request for Reexamination of the '416 Patent	12
V. IDENTIFICATION OF PRIOR ART AND SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY	12
A. Identification of Prior Art	12
B. Substantial New Questions of Patentability.....	13
1. <i>Baratti</i> raises an SNQ with respect to claims 1-18.	14

2.	The combined teachings of <i>Bauer</i> and <i>Wyman</i> raise an SNQ with respect to claims 1-18.	15
VI.	PROPOSED REJECTIONS OF THE '416 PATENT	16
A.	<i>Baratti</i> anticipates claims 1, 3-12, 14, and 16-18.	16
1.	Overview of <i>Baratti</i>	16
a.	Technical Field.....	16
2.	Independent Claim 1	19
	[1.1]: “A method of creating a clustered computing system, comprising:”	19
	[1.2a]: “storing license information for a computing cluster in a memory module associated with a computing node”	21
	[1.2b]: “wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster,”	23
	[1.2c]: “and one or more node license parameters that identifies a characteristic of a computing node;”	25
	[1.3]: “initializing the computing cluster in a first computing node;”	25
	[1.4]: “adding one or more available computing nodes to the computing cluster;”	26
	[1.5]: “and activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”	26
3.	Claim 3: “The method of claim 1, wherein the bundle-type parameter identifies at least one of a minimum or maximum node count threshold for the computing cluster.”	28
4.	Claim 4: “The method of claim 1, wherein: the node license parameters identify one or more bundle-types compatible with the computing node.”	28
5.	Claim 5: “The method of claim 1, wherein adding one or more available computing nodes to the computing cluster comprises assigning a computing node a new node license when the computing	

	node license is incompatible with a brand-type associated with the computing cluster.”	30
6.	Claim 6: “The method of claim 1, further comprising initiating a license check while the computing system is operating, wherein the license check evaluates at least one of:	32
	a license status of the computing node;	32
7.	Claim 7: “The method of claim 1, wherein the license check comprises implementing an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”	33
8.	Claim 8: “The method of claim 1, further comprising adding a computing node to the activated computing cluster.”	34
9.	Independent Claim 9	34
	[9.1]: “A computing system comprising at least two computing nodes, wherein at least one of the computing nodes comprises.”	34
	[9.2a]: “a cluster module stored on a non-transitory computer readable medium, ”	34
	[9.3a]: “a node module stored on a non-transitory computer readable medium,”	36
10.	Claim 10: “The computing system of claim 9, wherein the cluster module comprises a create cluster module to aggregate the at least two computing nodes into a clustered computing system.”	39
11.	Claim 11: “The computing system of claim 10, wherein the cluster module maintains a right to use license comprising licensing information for a computing cluster and a node license pool comprising licensing information for nodes in the computing cluster.”	39
12.	Claim 12: “The computing system of claim 10, further comprising an add node module to add a computing node to the clustered computing system.”	40
13.	Independent Claim 14	40

[14.1]: “A computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor,”	41
[14.2a]: “configure the processor to: store license information for a computing cluster in a memory module associated with a computing node,”	41
[14.2b]: “wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster, and”	41
[14.2c]: “one or more node license parameters that identifies a characteristic of a computing node;”	41
[14.3]: “initialize the computing cluster in a first computing node;”	41
[14.4]: “add one or more available computing nodes to the computing cluster; and”	42
[14.5]: “activate the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”	42
14. Claim 16: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to assign a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.”	42
15. Claim 17: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to initiate a license check while the computing system is operating, wherein the license check evaluates at least one of:	42
a license status of the computing node;	42
a brand-type associated with the computing node.”	42
16. Claim 18: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to implement an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”	43

B.	<i>Bauer</i> and <i>Wyman</i> render claims 1-12 and 14-18 obvious.	43
1.	Overview of <i>Bauer</i>	43
2.	Overview of <i>Wyman</i>	49
3.	Motivation to Combine <i>Bauer</i> and <i>Wyman</i>	53
4.	Independent Claim 1	55
	[1.1]: “A method of creating a clustered computing system, comprising:”	55
	[1.2a]: “storing license information for a computing cluster in a memory module associated with a computing node”	55
	[1.2b]: “wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster”	57
	[1.2c]: “and at least one additional bundle-type characteristic of the cluster,”	60
	[1.2d]: “and one or more node license parameters that identifies a characteristic of a computing node;”	61
	[1.3]: “initializing the computing cluster in a first computing node;”	63
	[1.4]: “adding one or more available computing nodes to the computing cluster;”	64
	[1.5]: “and activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”	66
5.	Claim 2: “The method of claim 1, wherein storing license information for a computing cluster in a memory module associated with a computing node comprises retrieving licensing information from an external licensing server.”	68
6.	Claim 3: “The method of claim 1, wherein the bundle-type parameter identifies at least one of a minimum or maximum node count threshold for the computing cluster.”	69
7.	Claim 4: “The method of claim 1, wherein: the node license parameters identify one or more bundle-types compatible with the computing node.”	70

8.	Claim 5: “The method of claim 1, wherein adding one or more available computing nodes to the computing cluster comprises assigning a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.”	71
9.	Claim 6 “The method of claim 1, further comprising initiating a license check while the computing system is operating, wherein the license check evaluates at least one of: a license status of the computing node; a node count in the computing cluster; and a brand-type associated with the computing node.”	72
10.	Claim 7: “The method of claim 1, wherein the license check comprises implementing an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”	74
11.	Claim 8: “The method of claim 1, further comprising adding a computing node to the activated computing cluster.”	76
12.	Independent Claim 9	76
	[9.1] “A computing system comprising at least two computing nodes, wherein at least one of the computing nodes comprises:”	76
	[9.2a] “a cluster module stored on a non-transitory computer readable medium,”	76
	[9.3a] “a node module stored on a non-transitory computer readable medium,”	79
	[9.4a] “and a management module stored on a non-transitory computer readable medium, the management module configured to manage compatibility operations with one or more external servers,”	82
13.	Claim 10: “The computing system of claim 9, wherein the cluster module comprises a create cluster module to aggregate the at least two computing nodes into a clustered computing system.”	83
14.	Claim 11: “The computing system of claim 10, wherein the cluster module maintains a right to use license comprising licensing information for a computing cluster and a node license pool comprising licensing information for nodes in the computing cluster.”	84

15.	Claim 12: “The computing system of claim 10, further comprising an add node module to add a computing node to the clustered computing system.”	84
16.	Independent Claim 14	85
	[14.1] “A computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor,”	85
	[14.2a]: “configure the processor to: store license information for a computing cluster in a memory module associated with a computing node,”	85
	[14.2b]: “wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and”	85
	[14.2c]: “at least one additional bundle-type characteristic of the cluster, and”	86
	[14.2d] “one or more node license parameters that identifies a characteristic of a computing node;”	86
	[14.3] “initialize the computing cluster in a first computing node;”	86
	[14.4] “add one or more available computing nodes to the computing cluster; and”	86
	[14.5] “activate the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”	86
17.	Claim 15: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to retrieve licensing information from an external licensing server.”	86
18.	Claim 16: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to assign a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.”	87
19.	Claim 17: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to initiate a license check while the computing system is operating, wherein the license check	

	evaluates at least one of: a license status of the computing node; a node count in the computing cluster; and a brand-type associated with the computing node.”	87
20.	Claim 18: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to implement an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”	88
C.	Ground 3: <i>Baratti</i> either anticipates claims 2 and 15, or it renders them obvious in view of <i>Bauer</i> or <i>Wyman</i>	88
1.	Claim 2: “The method of claim 1, wherein storing license information for a computing cluster in a memory module associated with a computing node comprises retrieving licensing information from an external licensing server.”	90
2.	Claim 15: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to retrieve licensing information from an external licensing server.”	90
D.	Ground 4: <i>Baratti</i> in view of <i>Hughes</i> renders obvious claim 13.	91
1.	Overview of <i>Hughes</i>	91
2.	Motivation to Combine <i>Baratti</i> and <i>Hughes</i>	92
3.	Claim 13: “The computing system of claim 10, wherein the management module provides a license key to and receives licensing information from an external licensing server.”	93
E.	Ground 5: <i>Bauer</i> and <i>Wyman</i> in view of <i>Hughes</i> renders obvious claim 13.....	94
1.	Overview of <i>Hughes</i>	94
2.	Motivation to Combine <i>Bauer</i> , <i>Wyman</i> , and <i>Hughes</i>	94
3.	Claim 13: “The computing system of claim 10, wherein the management module provides a license key to and receives licensing information from an external licensing server.”	96
VII.	CONCLUSION.....	96

I. RELATED PROCEEDINGS

A. Certification Regarding Statutory Estoppel

As required by 37 C.F.R. § 1.510(b)(6), NetApp certifies that the statutory estoppel provisions of 35 U.S.C. § 315(e)(1) and 35 U.S.C. § 325(e)(1) do not prohibit NetApp from filing the instant request for *ex parte* reexamination.

B. Other Proceedings

Valtrus and Key Patent Innovations Ltd. asserted the '416 patent in *Valtrus Innovations Ltd. v. NetApp Inc.*, Case No. 2:25-cv-00517 (E.D. Tex.). *See* L-1. They have also asserted the '416 patent in *Valtrus Innovations Ltd. v. FedEx Corp.*, Case No. 2:25-cv-00516 (E.D. Tex.). On June 27, 2025, the cases against NetApp and the FedEx defendants were consolidated. *See* L-2.

On July 25, 2025, Requester filed a declaratory judgment complaint involving the '416 patent in *NetApp, Inc. v. Valtrus Innovations Ltd. et al.*, 1:25-cv-06138 (S.D.N.Y. July 25, 2025). *See* L-3.

Petitioner is not aware of any other judicial or administrative matter that would affect, or be affected by, a decision in this proceeding.

C. Request for Expedited Reexamination

Due to the ongoing nature of the above-identified lawsuit, Requester respectfully urges that, pursuant to 35 U.S.C. § 305, this Request be granted and reexamination be conducted not only with “special dispatch,” but also with “priority over all other cases” in accordance with M.P.E.P. § 2261.

II. IDENTIFICATION OF CLAIMS THAT SHOULD BE REEXAMINED

Under 37 C.F.R. § 1.510(b)(2), NetApp respectfully requests *ex parte* reexamination of claims 1-18 of the '416 patent and the issuance of a reexamination certificate cancelling these claims.

III. BACKGROUND OF THE TECHNOLOGY

The '416 patent and all the prior art applied in this Request describe licensing processes for computer networks that employ “clusters,” collections of individual “nodes” (computer hardware that provide network services) that appear as single piece of hardware, for example, mass network storage, to the users of the network.

A. It was known before April 2006 that clustered networks provided the advantages of redundancy and flexible scalability.

Clustered computer networks were known when the '416 patent was filed in April 2006. It was known that cluster networks allowed network administrators to meet increased demands on the network by pooling resources into “clusters” that provide a service to network users. PA-2, [0003]; PA-1, [2:6-33] (“An obvious solution to ensure good availability would be to use well known clustering techniques. In the network data processing field, a cluster is a set of independent processors (nodes), connected over the network.”). A cluster avoids congestion problems by creating redundancy that distributes the work of providing the service across multiple computers instead of burdening a single computer. *Id.*

It was also known that clustered networks provided a flexible mechanism for scaling and dynamic resource allocation. When a cluster becomes overburdened, either because there are too many users accessing the services it provides or because a computer in the cluster malfunctioned, another computer can be added to the cluster. *E.g.*, PA-2, [0004]-[0005], [0048]; PA-1, [2:46-51].

Clustered networks thus provided a known solution to a common problem: how to dynamically scale the network and avoid service disruptions in response to spikes in network-user demand or the malfunctioning of a computer in the cluster. *Id.* It was also known that the concept of a cluster could be applied to a wide array of network devices and services. PA-2, [0005].

B. But it was also known that the flexibility inherent in clustered networks could wreak havoc on licensing management unless the system accounted for both node and cluster licensing information.

One side effect of the ability to flexibly add/remove nodes to/from a clustered network was that it could make licensing acquisition and management more difficult. PA-2, [0004]. For example, licenses typically restrict the number of computers in a cluster. *Id.*; PA-3, [20:20-64]. The network administrator thus had little flexibility to add new computers in response to, for example, a malfunctioning computer if the maximum number of licenses had already been reached. PA-3, [20:20-64]. One option for replacing a failed computer was for the administrator to use fewer than the maximum number of computers, setting aside one or more licensed computers as spares that could immediately be added to replace a failed computer. But this wasted money on a license that might not be needed and left the cluster operating at less than peak capacity. *Id.* Another option was to use all the licensed computers allowed for a cluster, but, when one failed, obtain a new license for a new computer. *Id.* But this resulted in overspending on a new license even though the total number of computers operating in the cluster remained unchanged. *Id.*

By April 2006 this problem had been solved using licensing management systems. *See generally* PA-1, PA-2, PA-3. These licensing management systems kept track of license information for both the computing cluster and the individual components that comprised the cluster. PA-1, [4:53-63]; PA-2, [0012]; PA-3, [11:54 - 14:62]. The licensing management systems tracked the number of computers in a cluster and whether they were licensed and then checked

how many computers the license required or permitted in the cluster. *See, e.g.*, PA-1, [5:4-16], PA-2, [0043]-[0045]. If the total number of computers was within the range permitted in the cluster, the cluster could provide cluster services. *See, e.g.*, PA-2, [0043]-[0045].

This enabled efficient, dynamic allocation of network resources to meet demand or to replace a failed resource. If a cluster's computer malfunctioned and had to be taken offline, the licensing management system would reduce the parameter representing the number of licensed computers in the cluster by one, freeing up space to add an additional computer. PA-3, [16:44-63]. After confirming that adding a computer to the cluster would not cause the number of computers in the cluster to exceed a maximum number, the licensing management system would permit the new computer to operate in the cluster and update licensing information indicating the number of computers in the cluster. PA-1, [5:37-45]; PA-2, [0045], [0051]-[0052]. So computers could be added to or removed from a cluster as circumstances demanded without unnecessary cost or effort. PA-1, [5:37-45]; PA-2, [0003]-[0006].

C. And it was known that the complexity of clustered networks required mechanisms to ensure nodes were compatible with each other and with the cluster.

The ability to flexibly add and remove elements from a clustered network required a way to ensure the compatibility of the nodes with the cluster. Otherwise, malicious users could add incompatible nodes to evade licensing restrictions (PA-1, [7:9-34]), or the incompatible nodes might not support licensed functionality (PA-3, [19:28-60]).

One way of preventing incompatibility between nodes and cluster was to incorporate node and cluster compatibility checks into the licensing-management system. PA-1, [7:9-34]; PA-3, [19:28:60]. This ensured not only that the nodes are licensed, but also that they are compatible with the cluster. PA-1, [7:9-34]; PA-3, [19:28-60].

These licensing checks required information about the node, like a unique node identifier, a node's function/purpose (PA-1, [4:60-62]), or operating system, hardware platform, product name (PA-3, [16:18-35]). They also required information about the clusters, like how many nodes are required (minimum) or permitted (maximum) in a cluster, a unique cluster identifier, and the cluster's intended function or purpose. PA-1, [4:55-62], [5:41-45]; PA-3, [20:20-64]. This process ensured each node and cluster was properly licensed, and that all the nodes in a cluster were compatible with each other and with the cluster they comprise. *Id.*; *see also* PA-1, [4:18-49], [6:8-7:33]. Often, confirmation of compatibility of the nodes with the cluster was required before a cluster's services were made available to the network. PA-1, [5:13-16], [6:24-43].

IV. OVERVIEW OF THE '416 PATENT AND ITS PROSECUTION HISTORY

A. Specification

The '416 patent relates to “compatibility enforcement in clustered computing systems.” Ex. A, 1:6-8. According to the patent, a “clustered computing system” is “formed from a group of loosely coupled computing devices that cooperate to provide the appearance that the devices function as a single system.” *Id.*, 1:11-14. Figure 2 illustrates a clustered computing system:

200

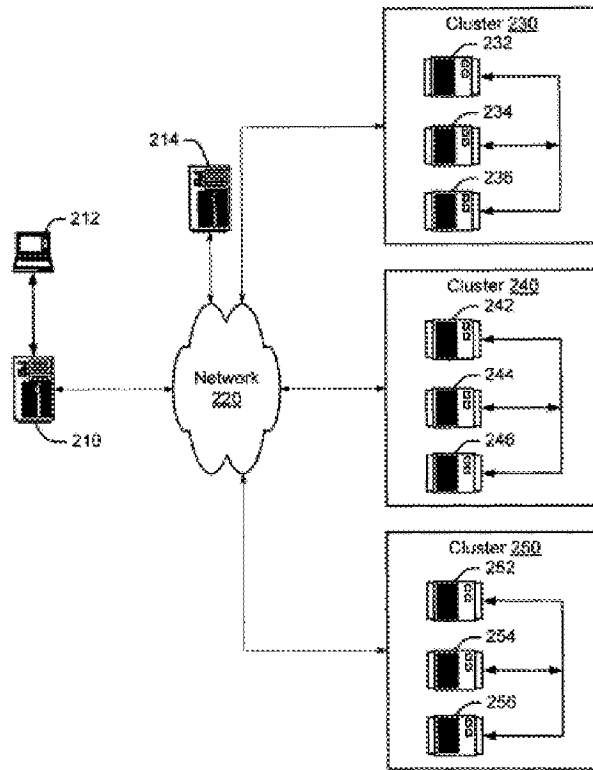


Fig. 2

Id., Fig. 2. The system includes “[s]torage network 200 compris[ing] a plurality of clusters 230, 240, 250, which may be implemented as one or more communicatively connected storage nodes” 232-256, each including “one or more storage devices such as, e.g., an array of storage devices.” *Id.*, 4:41-48.

The ’416 patent recognizes that clustered computing systems are not new, but contends prior systems had difficulty “ensuring compatibility between hardware, software and firmware operating on [the] clustered devices[,] and ensuring compliance with licensing terms associated with” the devices and cluster configurations. *Id.*, 1:24-29.

The '416 patent addresses this purported problem by implementing a licensing scheme with licensing requirements at both system and component levels. *Id.*, 5:40-50. The first step is “storing license information for a computing cluster in a memory module associated with a computing node.” *Id.* The license information includes “a bundle-type parameter that identifies a characteristic of the cluster and one or more node license parameters that identifies a characteristic of a computing node, initializing the computing cluster in a first computing node.” *Id.*

The '416 patent describes numerous bundle-type parameters that can characterize a cluster, including permissible cluster size (a minimum or maximum number of nodes) (*id.*, 5:50-52), the cluster's function or purpose (*id.*, 5:54-67), or a generic cluster “class” (*id.*, 8:1-5). Node license parameters that characterize a node include the node's function or purpose (*id.*, 5:66-67, 7:26-29), whether it can be added to a cluster it was not originally a part of (*id.*, 7:4-8), a node identifier (*id.*, 7:23-30), or a generic node “class” (*id.*, 8:1-5).

B. Claims 1-18 of the '416 Patent

This Request concerns all the '416 patent claims. The claims are reproduced below:

1. A method of creating a clustered computing system, comprising: storing license information for a computing cluster in a memory module associated with a computing node, wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster, and one or more node license parameters that identifies a characteristic of a computing node; initializing the computing cluster in a first computing node; adding one or more available computing nodes to the computing cluster; and activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.

2. The method of claim 1, wherein storing license information for a computing cluster in a memory module associated with a computing node comprises retrieving licensing information from an external licensing server.

3. The method of claim 1, wherein the bundle-type parameter identifies at least one of a minimum or maximum node count threshold for the computing cluster.

4. The method of claim 1, wherein: the node license parameters identify one or more bundle-types compatible with the computing node.

5. The method of claim 1, wherein adding one or more available computing nodes to the computing cluster comprises assigning a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.

6. The method of claim 1, further comprising initiating a license check while the computing system is operating, wherein the license check evaluates at least one of: a license status of the computing node; a node count in the computing cluster; and a brand-type associated with the computing node.

7. The method of claim 1, wherein the license check comprises implementing an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.

8. The method of claim 1, further comprising adding a computing node to the activated computing cluster.

9. A computing system comprising at least two computing nodes, wherein at least one of the computing nodes comprises: a cluster module stored on a non-transitory computer readable medium, the cluster module configured to manage compatibility operations associated with a computing cluster, the cluster module initializing a computing cluster in the first computing node,

the cluster module adding one or more available computing nodes to the computing cluster; a node module stored on a non-transitory computer readable medium, the node module configured to manage compatibility operations associated with the at least two computer nodes, the node module storing license information for the computing cluster, wherein the licensing information includes a bundle-type parameter that identifies a size of the computing cluster and at least one additional bundle-type characteristic of the computing cluster, and one or more node license parameters that identifies a characteristic of a first computing node; and a management module stored on a non-transitory computer readable medium, the management module configured to manage compatibility operations with one or more external servers, the management module activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.

10. The computing system of claim 9, wherein the cluster module comprises a create cluster module to aggregate the at least two computing nodes into a clustered computing system.

11. The computing system of claim 10, wherein the cluster module maintains a right to use license comprising licensing information for a computing cluster and a node license pool comprising licensing information for nodes in the computing cluster.

12. The computing system of claim 10, further comprising an add node module to add a computing node to the clustered computing system.

13. The computing system of claim 10, wherein the management module provides a license key to and receives licensing information from an external licensing server.

14. A computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor, configure the processor to: store license information for a computing cluster in a memory module associated

with a computing node, wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster, and one or more node license parameters that identifies a characteristic of a computing node; initialize the computing cluster in a first computing node; add one or more available computing nodes to the computing cluster; and activate the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.

15. The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to retrieve licensing information from an external licensing server.

16. The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to assign a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.

17. The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to initiate a license check while the computing system is operating, wherein the license check evaluates at least one of: a license status of the computing node; a node count in the computing cluster; and a brand-type associated with the computing node.

18. The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to implement an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.

.

C. Prosecution History

The application leading to the '416 patent (No. 11/411,732) was filed on April 26, 2006. Ex. B, at 25-29. There were a series of rejections during prosecution that Applicant responded to with arguments attempting to distinguish the prior art. In particular, the main reference applied during prosecution was *Bauer*. For instance, in response to a first office action rejection based on *Bauer* (US 2006/0036894 A1) (*id.* at 176), Applicant argued the claims differed from *Bauer* because it “describ[es] the activation of resources” not “adding . . . nodes to the computing cluster,” and it “does not disclose ‘a node module to manage compatibility operations associated with the at least two computer nodes.’” *Id.* at 207-208. These arguments were unsuccessful. *Id.* at 218-219.

Applicant did not materially advance prosecution until it amended the claims to require that the “bundle-type parameter” identified both “a size of the cluster and at least one additional bundle-type characteristic of the cluster.” *Id.* at 461-464, 468-473. Applicant acknowledged that *Bauer* discloses “the cluster resource status data structure includes a computer identifier field, an active resources field, an inactive resources field, and a number of licenses field,” but argued “none of the cluster resource data fields of *Bauer* constitute a bundle-type parameter.” *Id.* at 469-470.

The Examiner withdrew the outstanding rejections, but rejected the claims based on *Bauer* with two new references. *Id.* at 483-490. Applicant filed a Notice of Appeal and an Appeal Brief. *Id.* at 496, 500. In its brief, Applicant argued in part that *Bauer* failed to teach “activating a cluster based upon a size and an additional bundle-type parameter,” as required by the claims. *Id.* at 508-509. This time, the argument apparently worked. The Examiner conducted an interview with Applicant and agreed to allow the claims. *Id.* at 535-543.

D. Level of Ordinary Skill in the Art

A person having ordinary skill in the art at the time the purported invention was made (“ordinary artisan”) would have had at least a bachelor’s degree in electrical engineering, computer science, or similar field and two to three years of experience designing or analyzing computer storage systems. Additional relevant work experience can compensate for less education, and vice versa. Ex. C, ¶18.

E. Claim Construction for This Request for Reexamination of the ’416 Patent

“During reexamination, as with original examination, the PTO must give claims their broadest reasonable construction consistent with the specification.” *In re ICON Health Fitness, Inc.*, 496 F.3d 1374, 1379 (Fed. Cir. 2007).

V. IDENTIFICATION OF PRIOR ART AND SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

A. Identification of Prior Art

The application leading to the ’416 patent was filed April 26, 2006. The ’416 patent does not claim priority to any other applications.

Reexamination is requested in view of the following prior art references:

PA-1: U.S. Patent No. 6,574,612 to Baratti et al. (“*Baratti*”)

PA-2: U.S. Patent Application Publication No. 2006/0036894 A1 to Bauer (“*Bauer*”).

PA-3: U.S. Patent No. 5,260,999 to Wyman (“*Wyman*”).

PA-4: U.S. Patent Application Publication No. 2001/0044782 to Hughes et al. (“*Hughes*”).

PA-5: RS/6000 43P Model 140 Product Description (“*Workstation Description*”).

These prior art documents are listed on a form PTO/SB/08 filed concurrently herewith. Copies of all the listed references are attached.

Claims 1-18 are unpatentable in view of the following grounds:

Ground	Claims	Basis	Prior Art
1	1, 3-12, 14, 16-18	§102	<i>Baratti</i>
2	1-12, 14-18	§103	<i>Bauer</i> in view of <i>Wyman</i>
3	2, 15	§§ 102 or 103	<i>Baratti</i> alone or <i>Baratti</i> in view of <i>Bauer</i> or <i>Wyman</i>
4	13	§103	<i>Baratti</i> in view of <i>Hughes</i>
5	13	§103	<i>Bauer</i> in view of <i>Wyman</i> and further in view of <i>Hughes</i>

B. Substantial New Questions of Patentability

For an SNQ to be present, “it is only necessary that: (A) an item of information raises a[n] SNQ regarding at least one claim, i.e., the teaching of the item of information is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable; and (B) the same question of patentability . . . has not been decided by the Office in an earlier concluded examination or review of the patent, decided in a final holding of invalidity (after all appeals) by a federal court in a decision on the merits involving the claim, or raised to or by the Office in a pending reexamination or supplemental examination of the patent.” M.P.E.P. § 2816.02(I); *see also In re Vivint, Inc.*, 14 F.4th 1342, 1349 (Fed. Cir. 2021). “For example, the same question of patentability may already have been decided if the Office has previously considered, in an earlier concluded examination or review of the patent, the same question of patentability based on the same prior art.” *See* M.P.E.P. § 2816.02(I) (citing *In re Recreative Techs. Corp.*, 83 F.3d 1394 (Fed. Cir. 1996)).

Each proposed rejection in this Request presents a distinct SNQ of patentability. First, a reasonable Examiner would find the teachings of *Baratti* and *Wyatt*, submitted in this Request (as

also discussed in detail below), important in deciding whether the claims are patentable. Second, the same question of patentability has not been decided by the Office in an earlier concluded review or examination of the '416 patent. Although the Examiner did consider *Bauer* in the original prosecution, the Examiner lacked information that license management systems long-known in the art based licensing decisions on many more parameters than just the permitted size of a cluster. *See* Section IV.C. Third, the same question of patentability has not been decided in a final holding of invalidity by a federal court. Though the '416 patent is currently involved in co-pending district court litigations, those litigations have not resolved any question of validity regarding the '416 claims. *See supra* Section I.B. And finally, the same question of patentability has not been raised to or by the Office in a pending reexamination or supplemental examination. To NetApp's best knowledge, there has been no previous Office proceedings, *inter partes* review or *ex parte* reexamination, involving the claims of the '416 patent, much less any involving the specific combination of prior art that NetApp has raised below.

1. ***Baratti* raises an SNQ with respect to claims 1-18.**

As explained *supra* in Section IV.C, the '416 patent issued because the patent applicant convinced the Examiner that *Bauer's* system for licensing nodes in the clustered computer network failed to disclose "activating a cluster based upon a size and an additional bundle-type parameter," as required by the claims. Specifically, although the applicant conceded that *Bauer's* clustered network licensing system based activation on the *size* of the cluster, *Bauer* failed to teach using a *second* bundle-type parameter, such as the function or purpose of the bundle to determine whether a cluster should be activated. *Baratti*, like *Bauer*, teaches all the other elements of the claims but, unlike *Bauer*, *Baratti* expressly teaches using additional parameters, and, in particular, using a bundle-type parameter like a cluster identifier, to determine whether to activate the cluster.

2. The combined teachings of *Bauer* and *Wyman* raise an SNQ with respect to claims 1-18.

As explained *supra* in Section IV.C, the '416 patent issued after the patent applicant convinced the Examiner that *Bauer's* system for licensing nodes in the clustered computer network failed to disclose "activating a cluster based upon a size and an additional bundle-type parameter," as required by the claims. Specifically, although the applicant conceded that *Bauer's* clustered network licensing system based activation on the *size* of the cluster, *Bauer* failed to teach using a *second* bundle-type parameter, such as the function or purpose of the bundle to determine whether to activate the cluster.

Wyman remedies the deficiency of *Bauer* that led to the allowance of the '416 patent claims. Specifically, *Wyatt* describes a licensing system that takes into account numerous bundle-type parameters before activating a cluster, including parameters concerning the "node," "operating system," "specific hardware platform," or "duration of an allocation of license units." It would have been obvious to an ordinary artisan at the time of the alleged invention to modify *Bauer* in view of *Wyman*. Ex. C, ¶ 153. *Bauer's* system of enabling a cluster manager to reallocate licenses upon release would readily (and compatibly) utilize *Wyman's* licenses. *Id.* For example, *Bauer's* reallocation process enables a cluster manager to reallocate inactive licensable resources in response to a request, so long as the number of requested licensable resources is less than or equal to the number of licensed resources. *Wyman's* system similarly includes the ability to deallocate license units as needed, including upon receipt of a license release, freeing them up for use by other user nodes. *Bauer* and *Wyman* have processes to reallocate licensed resources based on abnormal termination of the use of the licensed resource.

VI. PROPOSED REJECTIONS OF THE '416 PATENT

A. *Baratti* anticipates claims 1, 3-12, 14, and 16-18.

1. Overview of *Baratti*

a. Technical Field

Baratti is directed to systems and methods for managing software licenses in distributed computing environments. PA-1, Abstract. It teaches a re-sizeable, high-availability licensing cluster that uses quorum requirements, cluster size restrictions, and node ids to provide centralized control over license authorization. *Id.*, [6:8-7:34].

As illustrated in Figure 5B of *Baratti*, the system comprises a plurality of client workstations (103) and a set of distributed license servers (101) organized into a cluster:

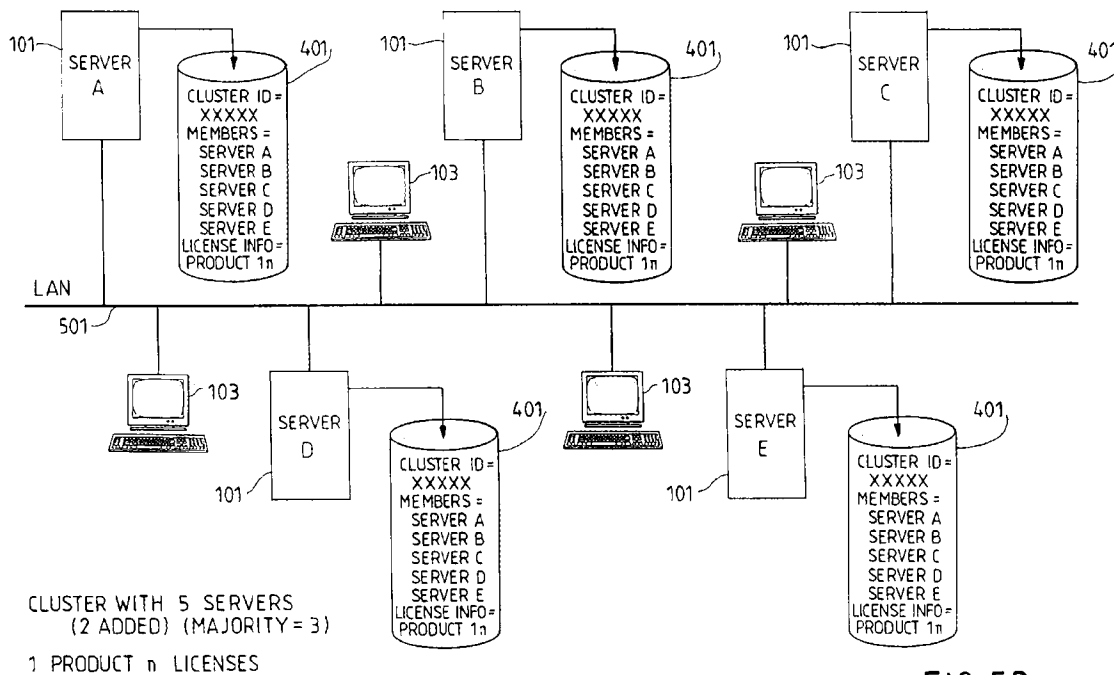


FIG. 5B

Each client has a local installation of the software but must obtain a license from the system before execution. PA-1, [4:19-37]. The license servers (101) manage licensing. *Id.* The preferred approach for the servers to license the clients is illustrated in Figure 2.

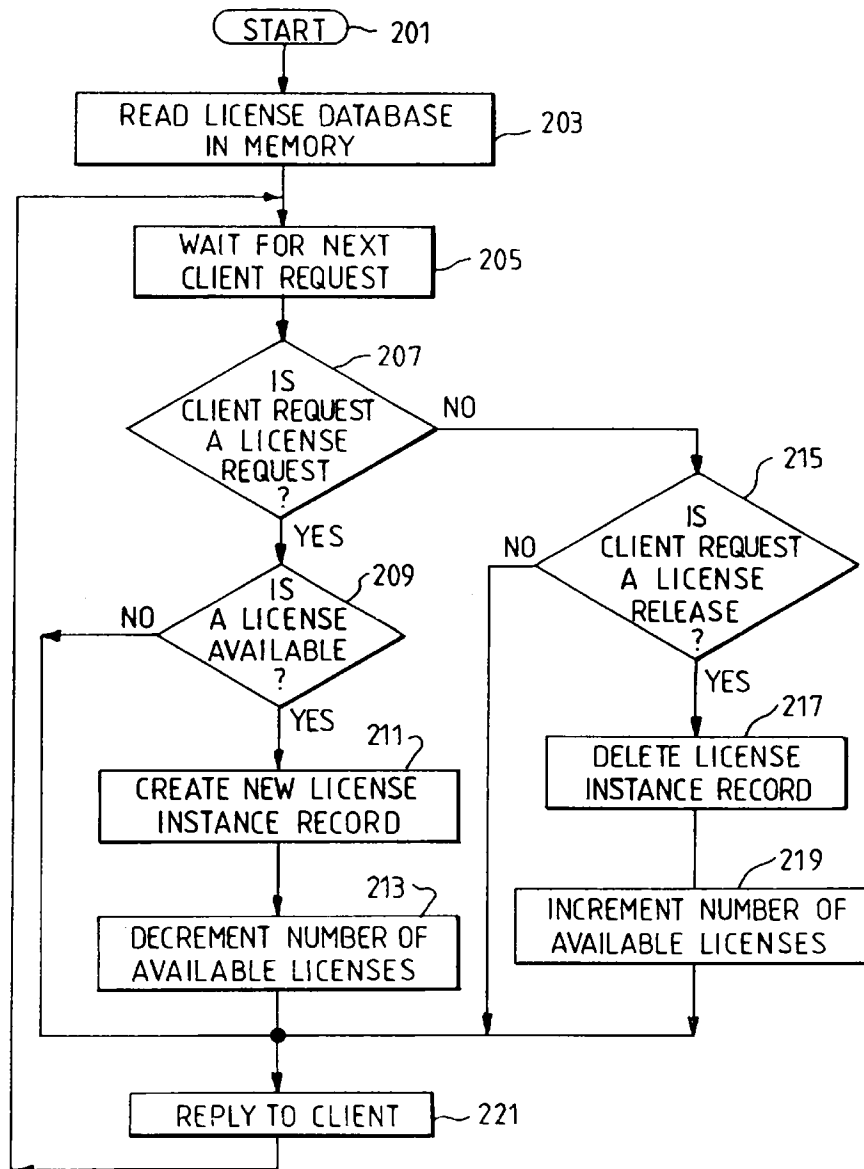


FIG. 2

The system is designed provide licensing services so long as a majority of the license servers in the cluster are active. PA-1, [5:4-16], [6:8-7:18]. This “majority-based consensus” model ensures fault tolerance in the event a server fails, while preventing unauthorized software usage through creation of additional, duplicate licensing clusters. *Id.*

Each server (101) in the cluster includes memory that stores licensing information. An exemplary licensing database includes the following information about the cluster and the individual servers (i.e., nodes, PA-1 [1:35-36]):

- the cluster ID;
- the total number of servers belonging to the cluster;
- the secure ID (possibly hardware-based) of each license server that has ever participated [in] the cluster;
- the software products managed by the system; and
- the number of available licenses for each product.

PA-1, [4:57-64].

Buratti introduces a mathematical constraint to enable re-sizing of the licensing cluster while preventing unauthorized software usage through creation of additional licensing clusters:

If we take S as the initial number of licence servers, $S1$ and $S2$ respectively as the minimum and the maximum number of servers that may belong to the cluster, the limitation is the following: the sum of the integer majority of $S1$ and of the integer majority of $S2$ must be strictly greater than $S2$. In other words the following rule must be fulfilled:

$$M(S1) + M(S2) > S2$$

with $S1 \leq S$ and $S2 \geq S$ and

where $M(x)$ is the integer majority of X

If the initial number *S* is odd the above rule is satisfied by limiting to 1 the number of servers that can be added or removed to the cluster.

PA-1, [6:13-6:26].

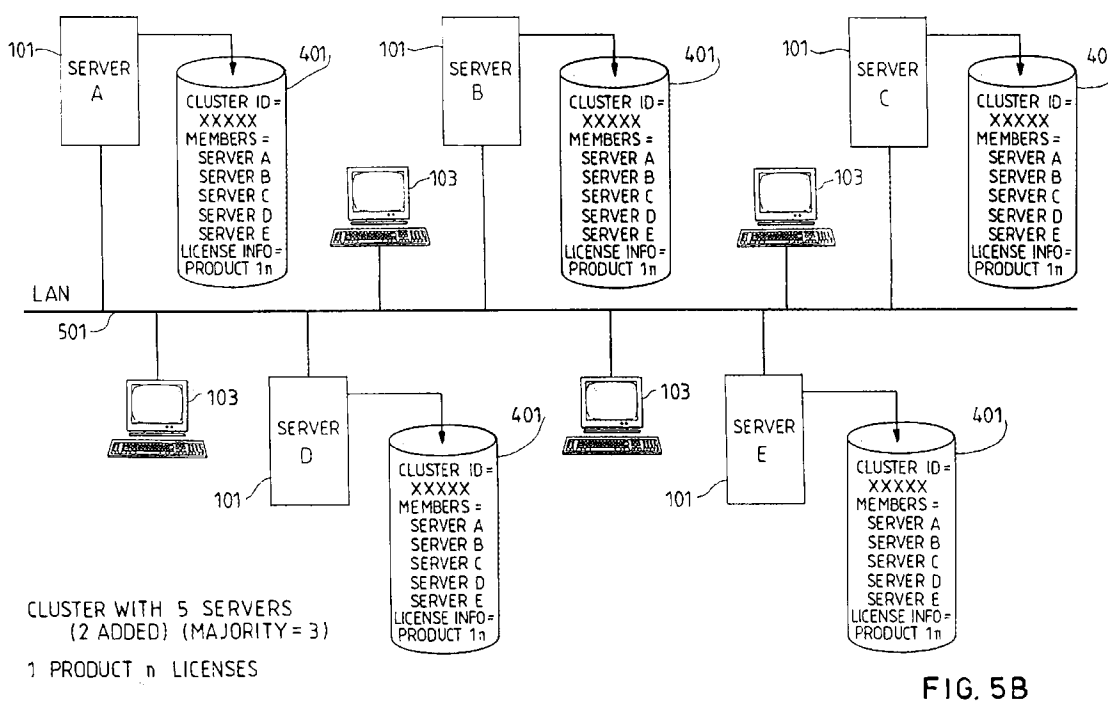
This constraint ensures that additional license servers can be added to the licensing cluster without enabling a malicious user to partition the licensing cluster and thereby fraudulently double the number of available licenses. PA-1, [5:46-6:42]. Although the preferred embodiment employs this constraint, *Baratti* also describes a clustered network that lacks this constraint, with the effect being that is less secure. *Id.*, [5:24-6:7].

The disclosed system offers several benefits, including: (1) scalability, through adjustment of license server count; (2) high availability, via redundant server architecture that prevents single points of failure; (3) security, through majority-based issuance that mitigates risks of unauthorized access; and (4) concurrency, enabling simultaneous software usage across multiple clients under controlled licensing.

2. Independent Claim 1

[1.1]: “A method of creating a clustered computing system, comprising:”

If the preamble is limiting, *Baratti* discloses it. Ex. C., ¶ 68. First, *Baratti* discloses the claimed “clustered computing system.” *Id.*



PA-1, Fig. 5B. This system allows client nodes 103 (which “could be any personal computer or workstation available on the market” PA-1, [4:14-16]), to use the services of a licensing cluster of server nodes 101, which “act as license servers” PA-1, [1:35-36]. *Baratti* explains:

An obvious solution to ensure good availability would be to use well known clustering techniques. In the network data processing field, a cluster is a set of independent processors (nodes), connected over the network. A cluster constitutes a sort of “black box” which provides certain services to end users. Like any ideal black box system, the end users do not need to know which node in the cluster they are connecting to.

Id., [2:17-24].

Second, *Baratti* also discloses a “method of creating a clustered computing system,” as claimed. Ex. C., ¶ 68. As explained in more detail below, *Baratti* describes methods of creating a clustered computing system by adding servers to a licensing cluster only if they meet the criteria established by the system. These criteria include both a minimum and a maximum number of servers in the licensing cluster, and server eligibility, such as whether it was part of the original

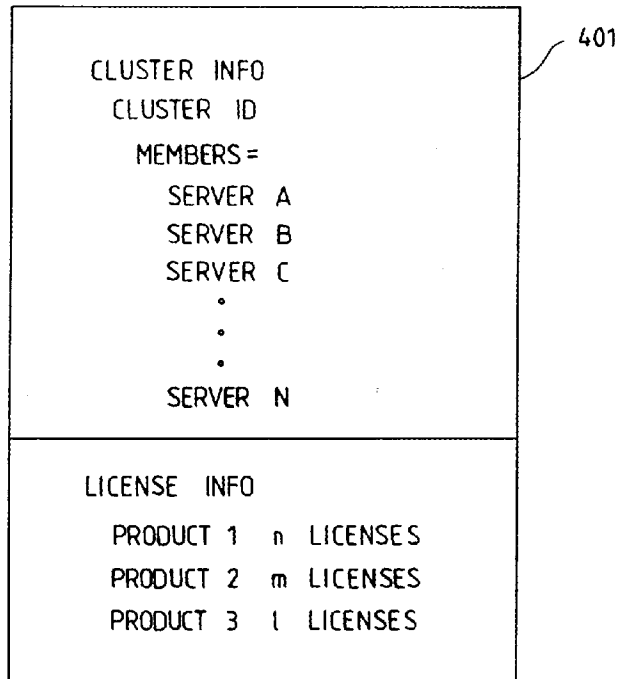
licensing cluster, which is determined through, among other mechanisms, a secure ID assigned to each server. PA-1, [4:53-63], [6:8-43], [6:65-7:33].

[1.2a]: “storing license information for a computing cluster in a memory module associated with a computing node”

Baratti discloses this limitation. Ex. C, ¶69. First, the servers (101) of *Baratti* are “nodes.” PA-1, [1:35-36] (“one or more of the nodes **101** act as license servers”). Ex. C, ¶ 69.

Second, *Baratti* describes at least one “memory module” “associated with a computing node,” as claimed. *Id.*, ¶ 70. *Baratti* teaches, for example, servers/nodes (101) that read licenses into license databases in server memory. PA-1, FIG. 2 (“Read License Database in Memory”), [4:18-25] (“The process starts at step **201** and goes straight to step **203** where the information about the software product to be licensed and the number of available licenses are read into the server memory.”) *Baratti* further teaches that each server/node (101) stores such a licensing database (401). PA-1, [4:53-55] (“FIG. 4 shows an example of license data base **401**, which, according to a preferred embodiment of the present invention, should be stored on each server **101**.”):

EXAMPLE OF LICENSE DATABASE



PA-1, Fig. 4. Ex. C, ¶ 70.

Last, *Baratti* discloses “storing license information for a computing cluster” in memory modules associated with a computing node (server). Ex. C, ¶ 71. *Baratti* explains:

The information contained in this data base is:

- the cluster ID;
- the total number of servers belonging to the cluster;
- the secure ID (possibly hardware-based) of each license server that has ever participated to the cluster;
- the software products managed by the system;
- the number of available licenses for each product.

PA-1, [4:55-63].

The data in these databases constitute “license information for a computing cluster,” because they are used by the licensing cluster for license management and define license cluster

and node characteristics, including the license cluster's minimum and maximum sizes, its function/purpose, a "cluster ID," nodes' secure IDs and the software products managed by the license cluster. *See, e.g., id.* Fig. 4; *see also, e.g.,* Ex. C, ¶¶ 71-72.

[1.2b]: "wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster,"

During prosecution, Applicant amended the claims adding the "one additional bundle-type characteristic of the cluster" language. Ex. B, at 468. Applicant stated that two different cluster characteristics are included in the "bundle-type parameter"; specifically, Applicant stated that "a bundle-type parameter is defined by the present specification as including size ranges and 'one or more additional parameters such as, e.g., an array, tape, computing, translating functional bundle.'" *Id.* Applicant then cited columns 5 and 6 of the '416 patent in support, which states that "clusters may be categorized into bundle types by one or more additional parameters such as, e.g., an array, tape, computing, translating functional bundle. For example, a bundle may refer to a sales bundle ... to meet a specific market or function." Ex. A, 5:50-61. That is, when the "at least one additional bundle-type characteristic" language was added to the claims, they required the claimed licensing information to include two different cluster characteristics: cluster size and another parameter, like cluster function or purpose, which would be relevant to the '416 patent's description of ensuring hardware compatibility by cluster type, like "array" or "tape." Ex. A, 5:43-50.

While Applicant contended no prior art teaches the use of two different cluster bundle-characteristics in licensing management, *Baratti* does. Ex. C, ¶ 74; Indeed, *Baratti*, exactly as the claims recite, considers at least three different cluster characteristics—cluster size ("the total number of servers belonging to the cluster"), cluster ID, and cluster function/purpose ("the

software products managed by the system”) when combining servers/nodes to create a licensing cluster. Ex. C., ¶ 74; *see supra* Section VI.A.2[1.2a].

Baratti further discloses a minimum set of servers that must be active for the license cluster to operate:

a minimum possible set must also be permanently determined. This minimum set is composed of the servers belonging to the cluster the first time the cluster reaches the minimum possible number of servers *s*₁; thereafter, these core servers cannot be de-activated any longer. Further changes in the cluster configuration can be done only by operating on the previously de-activated servers. Thus, let us suppose an initial cluster of 5 servers A, B, C, D and E. Server E is then de-activated (i.e. removed from the cluster). A, B, C and D must permanently belong to the cluster and cannot be removed any more. Previously de-activated server E and new server F can be added to the cluster and all future changes can only relate to E and F without touching the core set A, B, C and D.

PA-1, [7:19-34].

Baratti’s license information also includes a “at least one additional bundle-type characteristic of the cluster.” Ex. C, ¶¶ 75-76. For example, *Baratti* teaches the use of a cluster ID. PA-1, [4:57]. The claimed “additional bundle-type characteristic” also can be the function or purpose of the bundle. *See, e.g.*, Ex. A, 5:63-66 (explaining “bundles may have different characteristics within those listed above, for example, the purpose or function”). *Baratti*’s cluster-characterizing parameters also include information about the cluster’s function or purpose, which is defined by the software products it manages. PA-1, [4:62], Ex. C, ¶ 76.

The information above is stored in the license database 401, which is stored in every server/node of the cluster. Section VI.A.2[1.2a]. And it “includes a bundle type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster,” as claimed. Ex. C, ¶ 77.

[1.2c]: “and one or more node license parameters that identifies a characteristic of a computing node;”

Baratti discloses this limitation. Ex. C, ¶ 78. The '416 patent provides examples of “node license parameters that identif[y] a characteristic of a computing node,” including “a unique identifier for the node”. *Id.*, [6:35].

This same node characteristic is considered by *Baratti*. Ex. C, ¶ 80. *Baratti* stores each server's/node's “secure ID” in a “license data base 401,” which is stored in memory on every server/node. Section VI.A.2[1.2a].

[1.3]: “initializing the computing cluster in a first computing node;”

Baratti discloses this limitation. Ex. C, ¶ 81.

A cluster, comprised of nodes, cannot function without being initialized in at least one node. *Id.* *Baratti* discloses how servers, each one of which has been configured to store a licensing database (PA-1, [4:53-63]), are combined to form a cluster:

The network represented in FIG. 5a has three servers 101 serving a number n of licenses for software product 1 to a plurality of N clients 103. Each server 101 has a data base file 401 containing all the information described above. From the client point of view it does not make any difference whether a license is granted by server A, B or C. Any server 101 of the cluster can provide a license to any client 103.

PA-1, [5:4-9].

According to a preferred embodiment of the present invention, once a licence server has been added to the cluster, its unique identification is stored in the license data base of each server in the cluster and can never be deleted, unless the whole cluster and its identifier is deleted; this is to ensure that license servers cannot be replaced in the cluster. The license servers can only be added within the limits defined by the above rule and then deactivated to reduce the number of members of the cluster whose majority is to be up and running in order for the cluster to work.

PA-1, [6:65-7:7].

As such, *Buratti* discloses “initializing a cluster in a first computing node,” as claimed. Ex. C, ¶ 82.

[1.4]: “adding one or more available computing nodes to the computing cluster;”

Baratti discloses this limitation. Ex. C, ¶ 83. A cluster comprised of nodes necessarily had nodes added to it. *Id.*, ¶ 84. *Baratti*, in Figures 5a-5c, discloses a clusters with at three or five servers/nodes, which the ordinary artisan would have understood requires adding a first and then subsequent nodes to form the cluster. *See, e.g.*, PA-1, [6:66-7:9] (describing adding “a license server” and then “license servers” to the cluster). *Baratti* also describes how servers can be added back to a cluster after being removed from it. PA-1, [7:27-34] (“[L]et us suppose an initial cluster of 5 servers A, B, C, D and E. Server E is then de-activated (i.e. removed from the cluster). A, B, C and D must permanently belong to the cluster and cannot be removed any more. Previously de-activated server E and new server F can be added to the cluster and all future changes can only relate to E and F without touching the core set A, B, C and D.”).

Baratti’s adding of original, reactivated, and new servers to the cluster is consistent with the ’416 patent’s description of adding nodes. Ex. A, 9:8-13; *see* Ex. C, ¶ 85.

[1.5]: “and activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”

Baratti discloses this limitation. Ex. C, ¶ 86. The ’416 patent equates “activation” of a cluster with proceeding with the normal or intended operations of the cluster. Ex. C, ¶ 87. For example, the ’416 patent describes ensuring that “the cluster includes the minimum number of nodes before initializing the cluster services and proceed with normal operations.” Ex. A, 8:20-22.

Baratti similarly describes activation as the provision of cluster services. Ex. C, ¶ 88. According to *Baratti*, it is well-known and conventional to design clustered networks where a

cluster will not become active or remain active unless and until the number of servers in the cluster meets a required threshold:

An obvious solution to ensure good availability [of software licenses] would be to use well known clustering techniques. . . . However, . . . [a] redundant license server cannot simply take over the amount of licenses, served by another failing server; it must also ensure that, in no circumstances, the total number of licenses concurrently served can exceed the total number of available licenses, stored into the license authorization record.

Solutions to this problem [of managing licensing], based on a method called “majority” or “quorum”, are known, in which a certain number of license servers are configured to work cooperatively. As long as the majority of those servers is up and running and communicating with each other, all licenses are available, whereas as soon as the number of active license servers becomes less than the majority, all of the servers stop serving licenses.

PA-1, [2:17-41].

The system claimed in *Baratti* employs this technique. *See, e.g.*, PA-1, [2:66-3:8] (“in a network comprising . . . a cluster comprising an initial plurality of S license servers, a license management system . . . the license management system requiring that at least the integer majority of the plurality of license servers in the cluster is active at any time”). *Baratti* further teaches the additional bundle-type parameter of a minimum set of license servers that must be present in the licensing cluster in order for the licensing cluster to operate. PA-1, [7:19-34]. Moreover, the cluster can become active only when each server’s secure ID is confirmed (*id.*, [6:65-7:19]) and it is capable of licensing the appropriate software (*id.*, [4:53-63]).

Thus *Baratti* discloses activating the licensing cluster when it includes a number of licensing servers that complies with the quorum requirement (that identifies the size of the cluster) and the minimum set of licensing servers, and each server’s identity and function is confirmed.

Ex. C, ¶ 89.

3. Claim 3: “The method of claim 1, wherein the bundle-type parameter identifies at least one of a minimum or maximum node count threshold for the computing cluster.”

Baratti discloses the additional limitations of claim 3. Ex. C, ¶ 90. In particular, *Baratti* describes setting minimum and maximum values for the number of servers that can be in a cluster, which will then be used to determine how many servers can be added or removed from cluster for the cluster to become or remain active:

If we take S as the initial number of licence servers, S_1 and S_2 respectively as the minimum and the maximum number of servers that may belong to the cluster, the limitation [for activating the cluster or keeping it active] is the following: the sum of the integer majority of S_1 and of the integer majority of S_2 must be strictly greater than S_2 .

PA-1, [6:13-19].

This process identifies both a maximum and minimum node count threshold for the cluster. Ex. C, ¶ 91.

4. Claim 4: “The method of claim 1, wherein: the node license parameters identify one or more bundle-types compatible with the computing node.”

Baratti discloses the additional limitations of claim 4. Ex. C, ¶ 92. As noted, one example of a “bundle type” in the ’416 patent is its hardware type: “an array, tape, computing, translating functional bundle.” Ex. A, 5:53-57. In *Baratti*, the nodes are defined as “computing” type, i.e., servers, or, more specifically “IBM RISC System/6000 43P-140” and desktop computers or, more specifically, “IBM Personal Computer 300 GL.” PA-1, [4:11-17].

A bundle type may also be the function or purpose of the cluster or node. Ex. C, ¶ 93. *See supra* Section VI.A.2[1.2b]. *Baratti* discloses using information about the cluster’s function and purpose to ensure all nodes are compatible with it. PA-1, [4:56-63], [6:13-19]. Specifically, every

server in a cluster must be capable of licensing the software that the cluster has been created to license:

An obvious solution to ensure good availability would be to use well known clustering techniques. In the network data processing field, a cluster is a set of independent processors (nodes), connected over the network. A cluster constitutes a sort of “black box” which provides certain services to end users. Like any ideal black box system, the end users do not need to know which node in the cluster they are connecting to.

PA-1, [2:17-24].

The network represented in FIG. 5a has three servers 101 serving a number n of licenses for software product 1 to a plurality of N clients 103. Each server 101 has a data base file 401 containing all the information described above. From the client point of view it does not make any difference whether a license is granted by server A, B or C. Any server 101 of the cluster can provide a license to any client 103.

PA-1, [5:4-9].

Moreover, each server has a unique secure ID that is used to ensure that it belongs in the cluster, *i.e.*, that is compatible with the cluster’s security scheme:

According to a preferred embodiment of the present invention, *once a licence server has been added to the cluster, its unique identification is stored in the license data base of each server in the cluster and can never be deleted, unless the whole cluster and its identifier is deleted*; this is to ensure that license servers cannot be replaced in the cluster. The license servers can only be added within the limits defined by the above rule and then deactivated to reduce the number of members of the cluster whose majority is to be up and running in order for the cluster to work. Once the limit specified above has been reached, no new server can be added to the cluster, but previously deactivated servers must be used. *Allowing removal of license servers from the cluster or substitution would break the security of the cluster itself; for the same reason the unique ID of the license servers that initially form the cluster must be specified at cluster creation time.* When a server is removed from the cluster (de-activated) this information is not lost, because the server is just marked as no longer active in the cluster. In this way,

once the maximum number of servers is reached the whole set of servers that can ever be part of the cluster is definitely determined.

According to a preferred embodiment, to further increase the safety of the cluster, a minimum possible set must also be permanently determined. This minimum set is composed of the servers belonging to the cluster the first time the cluster reaches the minimum possible number of servers S_1 ; thereafter, these core servers cannot be de-activated any longer. Further changes in the cluster configuration can be done only by operating on the previously de-activated servers. Thus, let us suppose an initial cluster of 5 servers A, B, C, D and E. Server E is then de-activated (i.e. removed from the cluster). A, B, C and D must permanently belong to the cluster and cannot be removed any more. Previously de-activated server E and new server F can be added to the cluster and all future changes can only relate to E and F without touching the core set A, B, C and D.

PA-1, [6:65-7:33].

Thus, individual node characteristics constituting bundle types, like the type of node, the programs it licenses, and its unique secure ID, are contained in the licensing data base along with cluster characteristics, such as the programs the cluster licenses and a unique cluster ID, are used to ensure compatibility. *See supra* Sections VI.A.2 [1.2b], [1.2c], [1.5]; Ex. C, ¶¶ 94-95.

- 5. Claim 5: “The method of claim 1, wherein adding one or more available computing nodes to the computing cluster comprises assigning a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.”**

Baratti discloses the additional limitations of claim 5. Ex. C, ¶ 96.

The purpose of the servers in *Buratti* is to license software to the client computers on the network, which also are “nodes”:

A license management system allows a user to install a copy of a software program on N nodes of a network, but acquire only a limited number n licenses, where at any time, only the maximum number n copies of that program can be simultaneously run on the network.

* * *

In a typical network of interconnected computers with a license management system, as illustrated in FIG. 1, one or more of the nodes 101 act as license servers, while a plurality of nodes 103 act as clients of the license servers. The service provided by a license server 101 to its client 103 is that of granting or denying permission to run a given software program, according to the availability of a license record in the license server data base, and to the terms and conditions encoded in the license record itself.

PA-1, [1:17-42].

To perform the licensing management function, each server in the cluster must be configured to perform it, which requires, among other things, storing licensing records and other information in a database on the server:

The license server usually creates and stores license records in the license data base upon processing license certificate files, which are provided by the software vendor and complement the software program to which they are related. This license data base must be locked in some way to the specific instance of the license server (hardware + software) to prevent malicious users from copying the license data base to another license server machine and multiplying by two the number of licenses for all the software products contained in the license data base.

PA-1, [1:41-52].

Baratti describes a technique for adding new servers to cluster. PA-1, [7:19-34] (describing how and under what conditions a “new server” can be added.). Each server in the cluster must store, among other licensing data, information about “the software products managed by the system,” i.e., the licenses that each server in the cluster manages. PA-1, [4:62], [Figs. 2 and 3] [4:18-49]. Until the server’s memory is configured to store this license data it is not compatible with the cluster. Only after its database is updated to include this information will the server be compatible, and only then will it operate. Ex. C, ¶¶ 97-98.

Thus, *Baratti* teaches or at least renders obvious a process for obtaining “a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.” Ex. C, ¶ 98.

6. Claim 6: “The method of claim 1, further comprising initiating a license check while the computing system is operating, wherein the license check evaluates at least one of:

a license status of the computing node;

a node count in the computing cluster; and

a brand-type associated with the computing node.”

Baratti discloses this limitation. Ex. C, ¶ 99. *Baratti* describes a license check that is performed on the system after it has been constructed and begun to operate that evaluates the node count in the cluster:

If the initial number [of license servers] *S* is odd the above rule is satisfied by limiting to 1 the number of servers that can be added or removed to the cluster.

If the initial number *S* is even the rule is satisfied if either maximum 2 servers are added to the cluster OR 2 servers are removed from the cluster. These two conditions are mutually exclusive: this means that the system administrator can choose either to increase the initial server number by one or two, or to decrease the initial number by one or two. Once the choice is done the initial number will be the minimum or the maximum limit, respectively. As an example, once a server is removed from such a cluster, the initial number *S* becomes the maximum possible number of servers in the cluster, and the only possible changes are to remove another server or to reintroduce the removed one. In a similar way if the first change to the number of servers in the cluster is an addition of another server, the initial number *S* becomes the minimum possible number of servers in the cluster and the only possible further changes are another addition or the removal of the initially added server.

PA-1, [6:24-42].

According to a preferred embodiment, to further increase the safety of the cluster, a minimum possible set must also be permanently determined. This minimum set is composed of the servers belonging to the cluster the first time the cluster reaches the minimum possible number of servers S1; thereafter, these core servers cannot be de-activated any longer. Further changes in the cluster configuration can be done only by operating on the previously de-activated servers. Thus, let us suppose an initial cluster of 5 servers A, B, C, D and E. Server E is then de-activated (i.e. removed from the cluster). A, B, C and D must permanently belong to the cluster and cannot be removed any more. Previously de-activated server E and new server F can be added to the cluster and all future changes can only relate to E and F without touching the core set A, B, C and D.

PA-1, [7:19-34].

These techniques “compris[e] initiating a license check while the computing system is operating, wherein the license check evaluates “a node count in the computing cluster.” Ex. C, ¶ 100.¹

7. Claim 7: “The method of claim 1, wherein the license check comprises implementing an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”

Baratti discloses the additional limitations of claim 7. Ex. C, ¶ 101. For example, *Baratti* describes a process that will not allow a new server to be added (*i.e.*, it inhibits operation of the server/node) when the specified limit for the number of servers has been reached. PA-1, [7:7-9]

¹ Patent owner might argue that the only “licensing” described in *Baratti* is the licensing of the individual client nodes. But that argument would be wrong. The system’s enforcement of node and cluster requirements also constitutes licensing management. Specifically, a “license” is just a legal term for having permission to do something. *See, e.g.*, <https://www.merriam-webster.com/dictionary/license> (defining “license” as “permission to act”); *Lulirama Ltd., Inc. v. Axxcess Broad. Servs., Inc.*, 128 F.3d 872, 879 (5th Cir. 1997) (“When the totality of the parties’ conduct indicates an intent to grant such permission, the result is a legal nonexclusive license.”). The system of *Baratti* permits, that is, licenses, the nodes and cluster to operate, but only under certain specified conditions. That is a license, and the act of confirming that those conditions are met is a “license check.”

(“Once the limit specified above has been reached, no new server can be added to the cluster”). *Buratti* also states that “as soon as the number of active license servers becomes less than the majority, all of the servers stop serving licenses.” PA-1, [2:39-41]; Ex. C, ¶ 101.

8. Claim 8: “The method of claim 1, further comprising adding a computing node to the activated computing cluster.”

Baratti discloses the additional limitations of claim 8. Ex. C, ¶ 102. For example, *Baratti* describes how new servers can be added to an active cluster and how previously active servers can be added to a cluster after being removed from it. PA-1, [7:27-34] (“[L]et us suppose an initial cluster of 5 servers A, B, C, D and E. Server E is then de-activated (i.e. removed from the cluster). A, B, C and D must permanently belong to the cluster and cannot be removed any more. Previously de-activated server E and new server F can be added to the cluster and all future changes can only relate to E and F without touching the core set A, B, C and D.”).

Baratti’s adding of original, reactivated, and new servers to the cluster is consistent with the ’416 patent’s description of adding nodes to an activated cluster. Ex. A, 9:8-13; *see* Ex. C, ¶ 103; *see also supra* Section VI.A.2[1.4].

9. Independent Claim 9

[9.1]: “A computing system comprising at least two computing nodes, wherein at least one of the computing nodes comprises:”

If the preamble is limiting, *Baratti* discloses it. Ex. C, ¶ 104.

Baratti describes a clustered computing system that comprises at least two servers/nodes. PA-1, Figs. 5a-5c; [2:17-24]; Ex. C, ¶ 105; *see also* Sections VI.A.2[1.1]-[1.3].

[9.2a]: “a cluster module stored on a non-transitory computer readable medium,”

Baratti discloses this limitation. Ex. C, ¶ 106.

The '416 patent explains that a cluster module “includes logic to manage cluster-level compatibility functions.” Ex. A, 6:14-16. A “logic module may be software or firmware stored in a computer readable medium.” *Id.* 6:8-11. Each server of *Baratti* includes software stored on a non-transitory computer readable medium, i.e., “a logic module” that manages “cluster-level compatibility functions and thus corresponds to the claimed “cluster module.” Ex. C, ¶ 107; PA-1, [Fig. 22]; *supra* Sections VI.A.2[1.2b], [1.2c], and [1.5]. Specifically, as explained below each of the steps that the claimed “cluster module” performs is described in *Baratti. Id.*

[9.2b]: “the cluster module configured to manage compatibility operations associated with a computing cluster,”

Baratti discloses this limitation. Ex. C, ¶ 108.

Baratti explains that every node (i.e., every server) has the software to manage compatibility operations associated with the cluster. For instance, each server has a unique secure ID that is used by it and the other servers in the cluster to ensure that is compatible with the cluster’s security scheme. PA-1, [6:65-7:33] (“once a licence server has been added to the cluster, its unique identification is stored in the license data base of each server in the cluster and can never be deleted, unless the whole cluster and its identifier is deleted”), *id.* (“Allowing removal of license servers from the cluster or substitution would break the security of the cluster itself; for the same reason the unique ID of the license servers that initially form the cluster must be specified at cluster creation time.”). Ex.C, ¶ 109.

Thus, *Baratti* teaches that its system has a cluster module configured to manage compatibility operations associated with a computing cluster. *See supra* Sections VI.A.2 [1.2b], [1.2c], [1.5], [4]; Ex. C, ¶ 110.

[9.2c]: “the cluster module initializing a computing cluster in the first computing node,”

This limitation is substantially similar to limitation [1.3]. Ex. C, ¶ 111. This limitation specifies that the method is performed by a “cluster module,” which is just software stored in the memory of the node. Ex. A 6:8-11. The nodes of *Baratti* (the servers) likewise function through software stored in memory on the server. PA-1, [1:34-54], [4:18-49]. And so *Baratti* discloses this limitation for the reasons discussed above. *Supra* Section VI.A.2[1.3]; Ex. C, ¶ 111.

[9.2d]: “the cluster module adding one or more available computing nodes to the computing cluster;”

This limitation is substantially similar to limitation [1.4]. Ex. C, ¶ 112. This limitation specifies that the method is performed by a “cluster module,” which is just software stored in the memory of the node. Ex. A, 6:8-11. The nodes of *Baratti* (the servers) likewise function through software stored in memory on the server. PA-1, [1:34-54], [4:18-49]. And so *Baratti* discloses this limitation for the reasons discussed above. *Supra* Section VI.A.2[1.4]; Ex. C, ¶ 112.

[9.3a]: “a node module stored on a non-transitory computer readable medium,”

Baratti discloses this limitation. Ex. C, ¶ 113.

The ’416 patent explains that a “node module” is “logic to manage node-level compatibility functions.” Ex. A, 6:37-38. It also states that “logic module may be software or firmware stored in a computer readable medium.” *Id.* 6:8-11. As discussed above, *Baratti* discloses that the software on every node (the servers) stored on a non-transitory computer readable medium is used for licensing management. Ex. C, ¶114; *Supra* Section VI.A.2[1.2a-c].

And, as shown in further detail below, each node of *Baratti* has software stored in its memory that corresponds to the claimed “node module” because each server manages node-level compatibility functions. Ex. C, ¶ 115; Ex. A, 6:37-38.

[9.3b] “the node module configured to manage compatibility operations associated with the at least two computer nodes,”

Baratti discloses this limitation. Ex. C, ¶ 116.

Baratti describes a process for managing compatibility operations associated with at least two computing nodes that are performed by logic stored in the server's memory. *Id.*, ¶ 117. Specifically, each server in a cluster has logic that makes it capable of licensing the same software as other servers in the cluster license. PA-1, [2:17-24]. The servers are thus compatible with each other. This process, which ensures the nodes are compatible with each other, is a compatibility operation associated with the nodes. Ex. C, ¶ 117.

[9.3c] “the node module storing license information for the computing cluster,”

This limitation is substantially similar to limitation [1.2a-c]. Ex. C, ¶ 118. This limitation specifies that the method is performed by a “node module,” which is just software stored in the memory of the node. Ex. A, 6:8-11. The nodes of *Baratti* (the servers) likewise function through software stored in memory. PA-1, [1:34-54], [4:18-49]. And so *Baratti* discloses this limitation for the reasons discussed above. *Supra* Section VI.A.2[1.2a-c]; Ex. C, ¶ 118.

[9.3d] “wherein the licensing information includes a bundle-type parameter that identifies a size of the computing cluster and at least one additional bundle-type characteristic of the computing cluster,”

This limitation is substantially similar to limitations [1.2b]. Ex. C, ¶ 119. This limitation specifies that the method is performed by a “node module,” which is just software stored in the memory of the node. Ex. A, 6:8-11. The nodes of *Baratti* (the servers) likewise function through software stored in memory. PA-1, [1:34-54], [4:18-49]. And so *Baratti* discloses this limitation for the reasons discussed above. *Supra* Section VI.A.2[1.2b]; Ex. C, ¶ 119.

[9.3e] “and one or more node license parameters that identifies a characteristic of a first computing node;”

This limitation is substantially similar to limitation [1.2c]. Ex. C, ¶ 120. This limitation specifies that the method is performed by a “node module,” which is just software stored in the memory of the node. Ex. A, 6:8-11. The nodes of *Baratti* (the servers) likewise function through software stored in memory. PA-1, [1:34-54], [4:18-49]. And so *Baratti* discloses this limitation for the reasons discussed above. *Supra* Section VI.A.2[1.2c]; Ex. C, ¶ 120.

[9.4a] “and a management module stored on a non-transitory computer readable medium, the management module configured to manage compatibility operations with one or more external servers,”

Baratti discloses this limitation. Ex. C, ¶ 121.

The ’416 patent states that “management module 450 may include an external management control module 452 and an external license server access module 454 that provide an interface with an external license server 460.” Ex. A, 6:64-67. *Baratti* expressly states that the servers used in its system may be servers sold under the name IBM RISC System/6000 43P-140. PA-1, [4:10-12]. Persons skilled in the art when *Baratti* was filed knew at that time that the IBM RISC System/6000 43P-140 server described in *Baratti* had logic that enabled it to be compatible with one or more external servers. Ex. C, ¶ 122. For example, IBM RISC System/6000 43P-140 servers could communicate over the internet using the widely known and used TCP/IP protocol. *Id.*² Such persons would have recognized that TCP/IP is a protocol that helps “manage compatibility operations” between computers over the internet (IP stands for “Internet Protocol”), such as a node on a local area network and an external server. Ex. C, ¶ 122.

² A brochure describing the IBM RISC System/6000 43P-140 server confirms, for example, that it “allows business to be securely conducted on the Internet” and that it has an “Ethernet port for networking.” PA-5, pp. 1-2. A person skilled in the art in the year 2000 would have understood that this necessarily means that this device has on-board logic that manages compatibility operations with other computers over the internet, including “external servers.” Ex. C, ¶ 122.

[9.4b] “the management module activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”

This limitation is substantially similar to limitation [1.5]. Ex. C, ¶ 123. This limitation specifies that the method is performed by a “management module,” which is just software stored in the memory of the node. Ex. A, 6:8-11. The nodes of *Baratti* (the servers) likewise function through software stored in memory. PA-1, [1:34-54], [4:18-49]. And so *Baratti* discloses this limitation for the reasons discussed above. *Supra* Section VI.A.2[1.5]; Ex. C, ¶ 123.

10. Claim 10: “The computing system of claim 9, wherein the cluster module comprises a create cluster module to aggregate the at least two computing nodes into a clustered computing system.”

Baratti discloses the additional limitation of claim 10. Ex. C, ¶ 124.

The '416 patent explains that a cluster module is a module that “includes logic to manage cluster-level compatibility functions.” Ex. A, 6:14-16. A “logic module may be software or firmware stored in a computer readable medium.” *Id.* 6:8-11. Each server/node of *Baratti* includes software stored on a non-transitory computer readable medium, i.e., “a logic module” that aggregates at least two nodes into a clustered computing system and thus corresponds to the claimed “create cluster module.” Ex. C, ¶ 125; PA-1, [6:66-7:9] (describing adding “a license server” and then “license servers” to the cluster); [7:27-34] (describing how servers can be added back to a cluster after being removed from it); Figs. 5a-5c.

11. Claim 11: “The computing system of claim 10, wherein the cluster module maintains a right to use license comprising licensing information for a computing cluster and a node license pool comprising licensing information for nodes in the computing cluster.”

Baratti discloses the additional limitation of claim 11. Ex. C, ¶ 126.

Baratti describes logic on the CHNs or management terminal that maintains a “right to use license” that permits nodes to be used in a cluster. PA-1, [6:66-7:9]; [7:19-34]; Figs. 5a-5c; Ex. C,

¶ 127. For example, clusters are not active, that is, there is no “right to use” them, unless and until licensing conditions are met. *See id.* The licensing conditions are determined in-part based on licensing information about the computer cluster, e.g., the minimum and maximum number of nodes or the cluster ID. PA-1, [4:53-63]; [7:19-34]. Licensing conditions are also dictated by “licensing information for nodes in the computing cluster,” such as each server’s/node’s secure ID and the software it licenses. *Id.* The secure ID is part of a “node license pool.” *Compare* Ex. A, 7:24-26 (“the node license pool may be updated to associate a unique identifier (e.g., a serial number) with the node 25 that is assigned the license.”).

12. Claim 12: “The computing system of claim 10, further comprising an add node module to add a computing node to the clustered computing system.”

This limitation is substantially similar to limitation [1.4]. Ex. C, ¶ 128. This limitation specifies that the method is performed by an “add node module,” which is just software stored in the memory of the node. Ex. A, 6:14-20. The nodes of *Baratti* (the servers) likewise function through software stored in memory. PA-1, [1:34-54], [4:18-49]. And so *Baratti* discloses this limitation for the reasons discussed above. *Supra* Section VI.A.2[1.4]; Ex. C, ¶ 128.

13. Independent Claim 14

The only difference between claims 1 and 14 is that claim 14 recites a computer program product stored on a non-transitory computer-readable medium and executed by a computer processor to perform the method recited in claim 1. Every server (node) of *Baratti* has logic defined by a computer program stored in the server’s memory and that executes on the servers processor (CPU). Ex. C, ¶ 129. Thus, the discussion of claim 1, including all its elements, is incorporated below into the discussion of claim 14 and all its elements in its entirety.

[14.1]: “A computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor,”

To the extent the preamble is limiting, *Baratti* discloses it for the same reasons provided *supra* in Section VI.A.2[1.1]. Ex. C, ¶ 130.

[14.2a]: “configure the processor to: store license information for a computing cluster in a memory module associated with a computing node,”

As discussed above, *Baratti* discloses “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor, configure the processor” to perform various steps. PA-1 Figs. 4, 5A-5C; *see supra*, Section VI.A.13; Ex. C, ¶ 131. The rest of this claim limitation (“store license information . . . associated with a computing node”) is substantially similar to limitation [1.1]. Ex. C, ¶ 131; For the reasons provided above, *Baratti* discloses this limitation. Ex. C, ¶ 131; *see supra* Section VI.A.2[1.1].

[14.2b]: “wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster, and”

This limitation is substantially similar to limitation [1.2b]. Ex. C, ¶132. For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.2[1.2b]; Ex. C, ¶ 132.

[14.2c]: “one or more node license parameters that identifies a characteristic of a computing node;”

This limitation is substantially similar to limitation [1.2c]. Ex. C, ¶133. For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.2[1.2c]; Ex. C, ¶ 133.

[14.3]: “initialize the computing cluster in a first computing node;”

This claim limitation is substantially similar to limitation [1.3]. Ex. C, ¶ 134. For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.2[1.3]; Ex. C, ¶ 134.

[14.4]: “add one or more available computing nodes to the computing cluster; and”

This claim limitation is substantially similar to limitation [1.4]. Ex. C, ¶ 135. For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.2[1.4]; Ex. C, ¶ 135.

[14.5]: “activate the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”

This claim limitation is substantially similar to limitation [1.5]. Ex. C, ¶136. For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.2[1.5]; Ex. C, ¶ 136.

- 14. Claim 16: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to assign a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.”**

As discussed for claim 14, *Baratti* discloses “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor, configure the processor” to perform various steps. Ex. C, ¶ 137. The rest of this claim limitation is substantially similar to claim 5. *Id.* For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.5; Ex. C, ¶ 137.

- 15. Claim 17: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to initiate a license check while the computing system is operating, wherein the license check evaluates at least one of:**
- a license status of the computing node;**
 - a node count in the computing cluster; and**
 - a brand-type associated with the computing node.”**

As discussed for claim 14, *Baratti* discloses “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by

a computer processor, configure the processor” to perform various steps. *See supra* Section VI.A.13; Ex. C, ¶ 138. The rest of this claim limitation is substantially similar to limitation [6.1], including that it requires evaluation of “a node count in the computing cluster.” Ex. C, ¶138. For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.6[6.1]; Ex. C, ¶ 138.

16. Claim 18: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to implement an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”

As discussed for claim 14, *Baratti* discloses “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor, configure the processor” to perform various steps. *See supra* Section VI.A.14; Ex. C, ¶ 139. The rest of this claim limitation is substantially similar to claim 7. Ex. C, ¶139. For the reasons discussed above, *Baratti* discloses this limitation. *Supra* Section VI.A.7; Ex. C, ¶ 139.

B. *Bauer* and *Wyman* render claims 1-12 and 14-18 obvious.

1. Overview of *Bauer*

Bauer discloses a system for managing the licensing of resources, such as processors, in clustered computing networks. PA-2, Abstract, [0017], [0039]. An exemplary computer system 100 that contains and/or is connected to licensable resources is depicted in Figure 1:

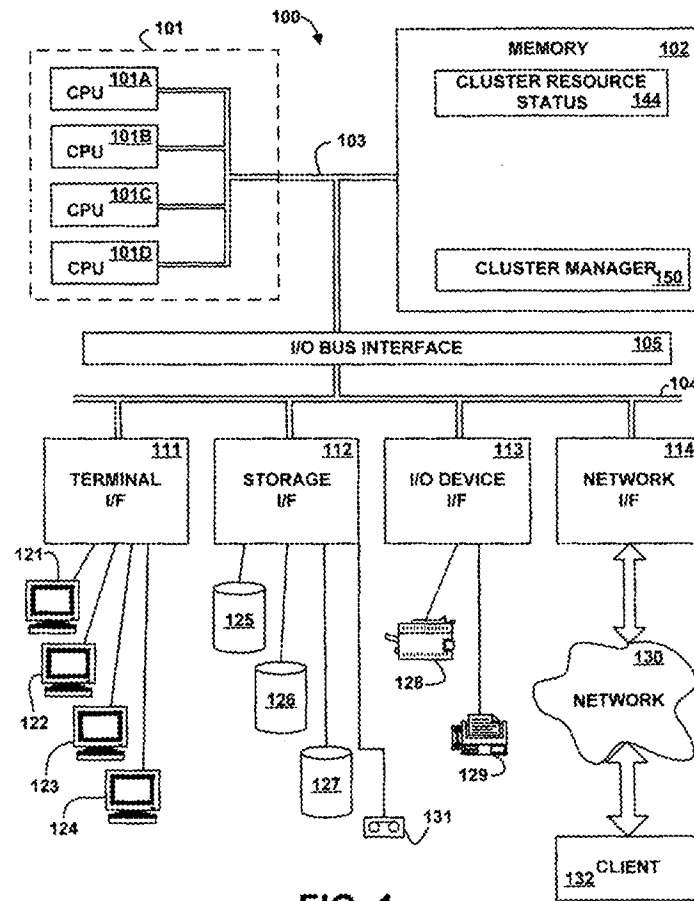


FIG. 1

Id., Fig. 1. Licensable resources in system 100 include processors 101 (101A-101D). PA-2, [0017]. But *Bauer* is not limited to licensing processors: “any appropriate resource may be licensed to the cluster,” including memory, queues, secondary storage, I/O adapters, I/O processors, network bandwidth, or other resources. *Id.*, [0019], [0039]. Secondary storage may be, for example, direct access storage devices (DASD) 125-127 connected through an I/O bus interface 105. *Id.*, [0019], [0020], [0039]. A cluster manager 150 manages the status of licensable resources using cluster resource status 144, both of which are depicted as included in memory 102. *Id.*, [0016]-[0018].

Computer system 100 is an example of a computer system in a cluster 200. Figure 2 depicts computer system 100 (e.g., computer system 100-1) in cluster 200:

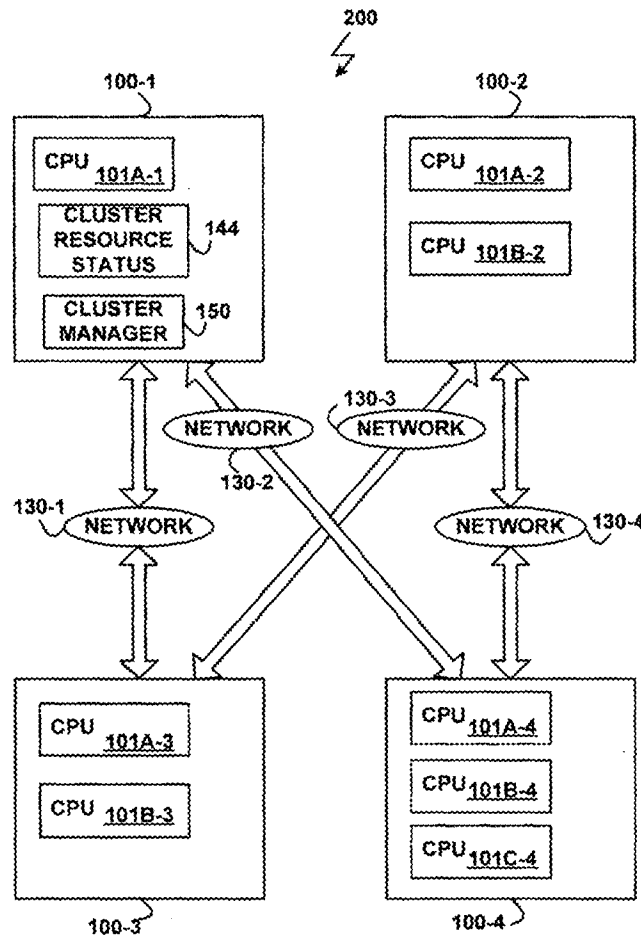


FIG. 2

Id., Fig. 2; *see also id.*, [0037] (noting that each of 100-1 through 100-4 are “referred to generically” as computer system 100). Resources that are active and licensed to the cluster (i.e., usable by systems in the cluster) are distinguished from resources that are inactive and unlicensed to the cluster (i.e., *not* usable by systems in the cluster). *Id.*, Fig. 3; *see also id.* [0042].

Computer system 100-1 can include a cluster resource status 144. The cluster resource status stores the status of each licensable resource at each computer system in cluster 200. *Id.*, [0018], [0040]. Cluster resource status 144 also includes a data structure that identifies resources that are active and licensed to the cluster (field 330), and resources in each computer system that are inactive and unlicensed to the cluster (field 335):

144

CLUSTER RESOURCE STATUS		
325 COMPUTER ID	330 ACTIVE RESOURCES	335 INACTIVE RESOURCES
COMPUTER A	CPU A	CPU B, C, D
COMPUTER B	CPU A, B	CPU C, D
COMPUTER C	CPU A, B	CPU C, D
COMPUTER D	CPU A, B, C	CPU D

305
310
315
320

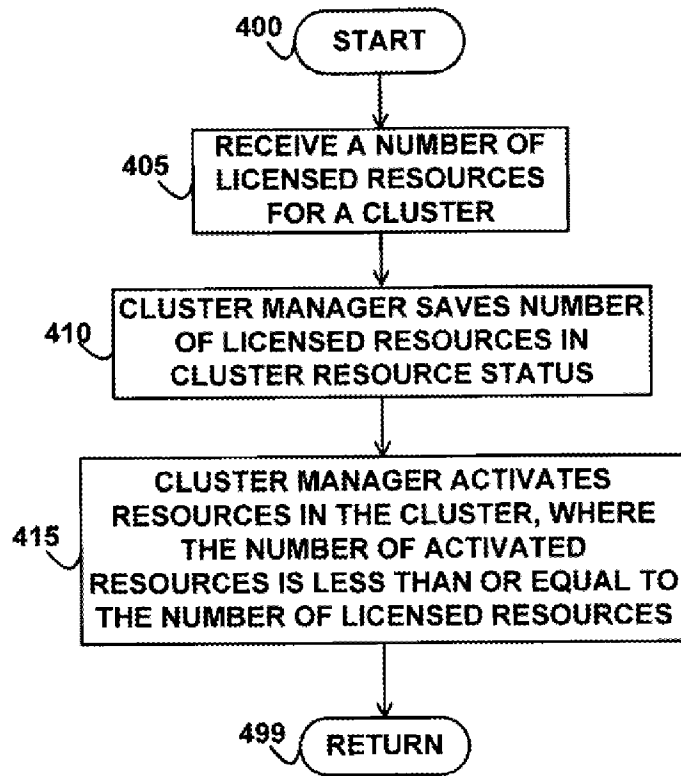
NUMBER OF
 LICENSES

340

FIG. 3

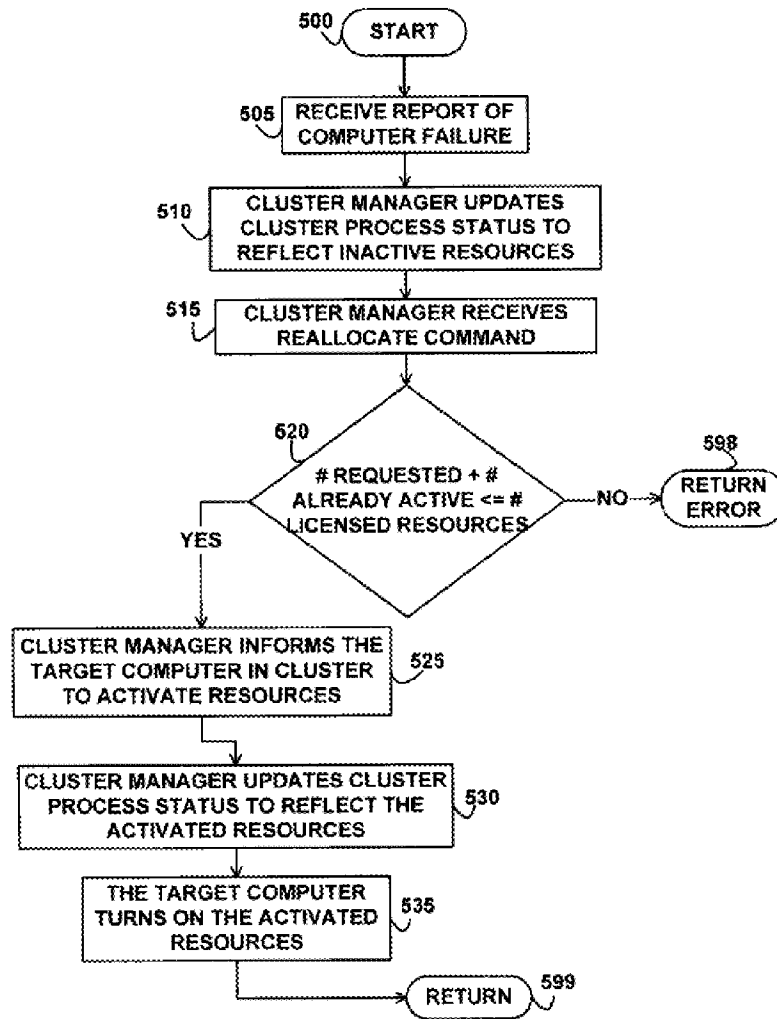
Id., Fig. 3. For example, Fig. 3 depicts CPU A in Computer A is active and licensed to the cluster, whereas CPUs B, C, and D, are not licensed for use by other systems in the cluster. *Id.*, Fig. 3; *see also id.*, [0041]-[0042]. An ordinary artisan would have understood other licensable resources, such as storage (e.g., Direct Attached Storage Devices 125-127) would be listed in cluster resource status 144 in a similar fashion. Ex. C, ¶¶ 140-142.

Cluster resources status 144 may also include a number of licenses field 340. PA-2, [0043]. This field 340 indicates the “maximum number of licensed resources available to the cluster 200.” *Id.* Figure 4 depicts the process for populating cluster resource status 144:



Id., Fig. 4. When a license is received (step 405), cluster manager 150 saves the number of licensed resources in the number of licenses field 340 (step 410). *Id.*, [0044]-[0045]. Cluster manager 150 then activates the licensed resources at the computers in the cluster 200, in compliance with the license (step 415). *Id.*, [0045]. Said another way, cluster manager 150 will activate resources, such that the number of activated resources (stored in cluster resource status 144) is less than or equal to the number of licensed resources (stored in number of licenses field 340). *Id.*, [0045]; Ex. C, ¶ 143.

Figure 5 depicts an operation of cluster manager 150 to reallocate licensable resources, such as after a failure. *Id.*, [0046].

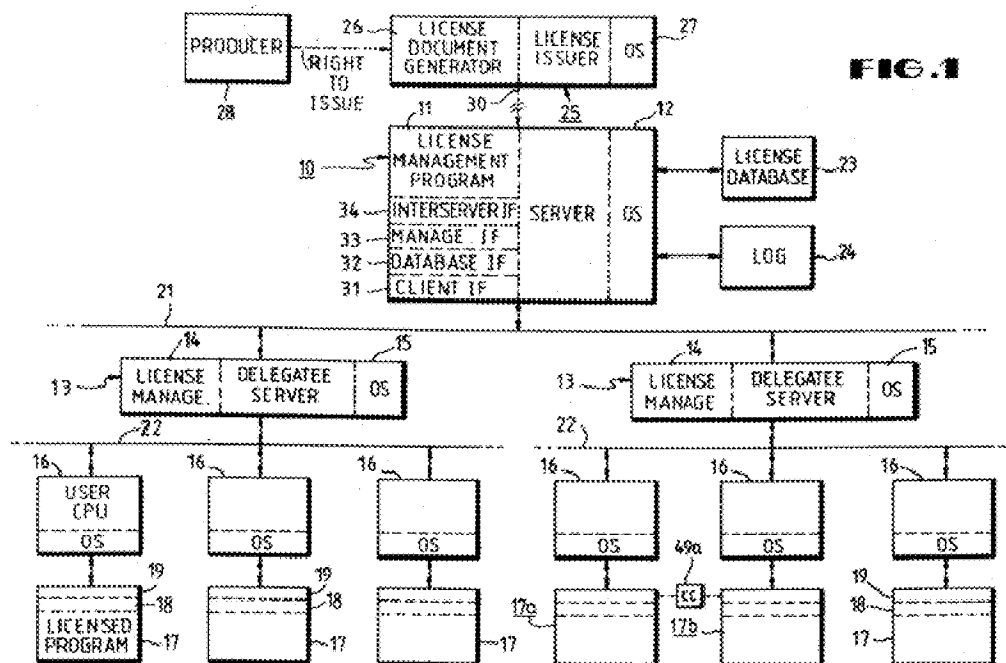


Id., Fig. 5. As shown, cluster manager 150 receives a reallocate command which requests activation of a number of requested resources (step 515). [0048]. Cluster manager 150 determines if activating the number of requested resources would exceed the number of licenses field 340 (in cluster resource status 144) (step 520). *Id.*, [0049]. If activating the requested resources would not exceed that number, then cluster manager 150 informs the target computer to activate the resources (step 525) and updates the cluster resource status 144 to account for activated resources (step 530). *Id.*, [0051]-[0052]. The target computer then activates the requested resources (step 535). *Id.*, [0052]; Ex. C, ¶ 144.

2. Overview of Wyman

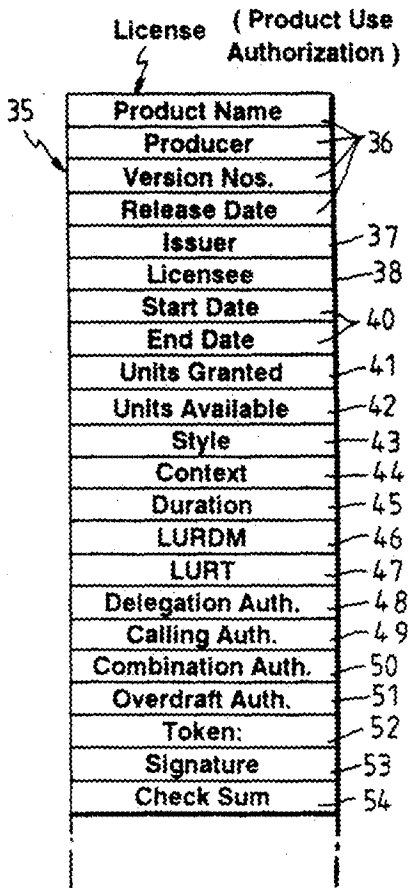
Wyman relates to software licensing management in a distributed computer system. PA-3, Abstract. *Wyman's* licensing management system is versatile and employable in different network architectures, including clustered computing networks. *Id.*, [6:61-65]; Ex. C, ¶ 145.

Figure 1 depicts a license management facility; this facility includes a license server 10 that utilizes a product use authorization (or license) issued by a license issuer 25. PA-3, [7:49-51], [9:25-30], [10:41-45].



Id., Fig. 1. When licensed program 17 is started, unit 18 is executed to make a request allocation call via stub 19 to determine whether a license is granted for the program. *Id.*, [23:3-11], Fig. 5 (steps 61-63). Ex. C, ¶ 146.

A license document, such as the license depicted in Figure 2, includes fields relating to the parameters of the license:



PA-3, Fig. 2. The license includes fields identifying the licensed product (36), the license issuer (37), start and end dates for the license (40), and the license units granted for the license (41). *Id.*, [12:12-34]. As license units are granted to users, the copy of the license stored on the license server 12 stores the number of available units (42). *Id.*, [12:34-38]; Ex. C, ¶ 147.

Fields 43-46 define more granular features of license 35, as shown in Figure 3 below:

43 Style	44 Context	45 Duration	46 LURDM
Allocative	Network	Transaction	Constant
Consumptive	Execution_Domain	Assignment	Table Lookup
Private	Login_Domain	Immediate	Private
	Node_ID		
	Process_Family		
	Process		
	User_Name		
	Product_Name		
	Operating_System		
	Platform_ID		
	Private		

FIG.3

PA-3, Fig. 3. License 35 can define a license style 43. *Id.*, [12:40-43], [15:18-22]. Like the license in *Bauer, Wyman* can define an “allocative” style of license that allocates a portion of the license for product 17 “at the time of activation” or “at the time that product use is enabled on a particular platform (user CPU 16).” *Id.*, [15:22-31]; *see* PA-2, [0047] (discussing reallocation of resources upon failure of a licensed resource). Also like *Bauer*, when the program is terminated or disabled, the allocated portion of the licensed product is deallocated, freeing it for use by others. PA-3, [15:32-35]; PA-2, [0051]-[0052]; Ex. C, ¶ 148.

License 35 can also define a context 44, which specifies the contexts in which a license can be used. PA-3, [16:18-35]. License data is allocated or used within a specific context, such as a “specific network,” “a node,” “an operating system,” or “a specific hardware platform.” *Id.*, [16:24-28].

License 35 can define a duration 45. Duration field 45 can define the “duration of an allocation of license units” to context 44. *Id.*, [16:36-39]. If duration 45 is set to “transaction,” this indicates that license units should be allocated upon request (to a specific context) and deallocated when they are released. *Id.*, [16:44-50].

And license 35 can define an LURDM 46. *Id.*, Fig. 3. LURDM, or “license unit requirement determination method,” stores information usable for calculating the number of units that should be allocated in response to a license request. *Id.*, [17:36-40]. One example of a LURDM field is “table lookup”—an exemplary License Unit Requirement Table (LURT) for such embodiments is depicted in Figure 4:

License Unit Requirements Table			
Row Selector	Columns		
Platform ID	A	B	C
PC-0	10	230	-1
PC-1	12	230	-1
VAX 6210	158	300	150

FIG. 4

Id., Fig. 4. In this example, each Platform ID defines a context in which a licensed program 17 is to be used along with the cost (in license units) for use of the licensed program 17 in that context. *Id.*, [17:57-18:19]. The columns concern products having “utility . . . highly sensitive to the characteristics of the specific platform on which they are run.” *Id.*, [18:2-5]. The intersection of the Platform ID and one of Columns A-C indicates “the number of units” required for licensing a particular product on a particular Platform ID. *Id.*, [17:36-40]. For example, the units required for a system of type PC-0 to utilize products in column A (10) are lower than the units required for a VAX 6210 type to utilize the same products (158). *Id.*, [18:2-19]. This shows, for example, different pricing tiers that reflect the relative utility of different platforms.

Wyman explains the format of the license depicted in Figures 2 and 3 provides the ability to create “[m]any different types of license [] using different combinations of contexts, duration and policy.” *Id.*, [20:5-10]. For example, by setting context 44 to a process ID, duration 45 to

“transaction,” and style 43 to “allocative,” it enables the creation of a “concurrent use” license that “limits the number of simultaneous uses of a licensed product.” *Id.*, [20:20-27].

3. Motivation to Combine *Bauer* and *Wyman*

It would have been obvious to an ordinary artisan at the time of the alleged invention to modify *Bauer* in view of *Wyman*. Ex. C, ¶ 153. Both *Bauer* and *Wyman* are from the same field of endeavor as the claimed invention, namely licensing for clustered computer networks. Moreover, each of *Bauer* and *Wyman* address the same problem faced by the '416 patent, namely how to efficiently allocate licenses to nodes in clustered computer networks. *Id.* *Bauer*'s system of enabling a cluster manager to reallocate licenses upon release would readily (and compatibly) utilize *Wyman*'s licenses. *Id.* For example, *Bauer*'s reallocation process enables cluster manager 150 to reallocate inactive licensable resources in response to a request, so long as the number of requested licensable resources is less than or equal to the number of licensed resources. PA-2, Fig. 5, [0048]-[0049]. *Wyman*'s system similarly includes the ability to deallocate license units as needed, including upon receipt of a license release, freeing them up for use by other user nodes. PA-3, [16:44-50], [22:11-40], Fig. 5 (box 69). *Bauer* and *Wyman* have processes to reallocate licensed resources based on abnormal termination of the use of the licensed resource. PA-2, Fig. 5 (reallocating licensed resources based on “computer failure”); PA-3, [16:50-52] (“Abnormal termination of the process or context having made the original license request will be semantically equivalent to a license release.”); *see also, e.g.*, Ex. C, ¶ 153.

In order for a reference to be proper for use in an obviousness rejection under 35 U.S.C. 103, the reference must be analogous art to the claimed invention. *In re Bigio*, 381 F.3d 1320, 1325 (Fed. Cir. 2004). A reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or

(2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). Note that "same field of endeavor" and "reasonably pertinent" are two separate tests for establishing analogous art; it is not necessary for a reference to fulfill both tests in order to qualify as analogous art. *See Bigio*, 381 F.3d at 1325. Ex. C, ¶ 154.

As Dr. Weissman explains in his declaration, *Wyman* is analogous art because it is both from the same field of endeavor of the '416 patent and *Bauer*, and because they all solve essentially the same problem: how to efficiently license nodes in clustered computer networks. Ex. C, ¶ 155.

An ordinary artisan would also find *Wyman*'s license fields as a predictable, easy-to-implement alternative to *Bauer*'s number of licenses field 340. *Id.*, ¶ 156. *Bauer* explains while the number of licenses field 340 lists the "maximum number of licensed resources" for cluster 200, that field can also include "separate numbers of licenses for different types of resources." PA-2, [0043]. *Bauer*'s existing extensibility for license data would be complemented by *Wyman*'s more extensible license that includes the number of license units, among other fields such as start/end date, context, and duration. Ex. C, ¶ 156. The ordinary artisan would have understood how to replace *Bauer*'s number of licenses field 340 with *Wyman*'s more comprehensive set of parameters, providing for additional functionality. *Id.* *Wyman* even calls its licenses an "improvement" to "cluster licenses," such as those used in *Bauer*, for it "provides a general mechanism which allows an effectively unlimited range of allocation scopes to be defined." PA-3, [25:44-53]; Ex. C, ¶ 156.

4. **Independent Claim 1**

[1.1]: “A method of creating a clustered computing system, comprising:”

If the preamble is limiting, *Bauer* discloses it. Ex. C, ¶ 157. *Bauer*’s cluster 200 of computer systems 100-1, 100-2, 100-3, and 100-4 teaches the claimed “clustered computing system.” PA-2, [0037], Fig. 2.

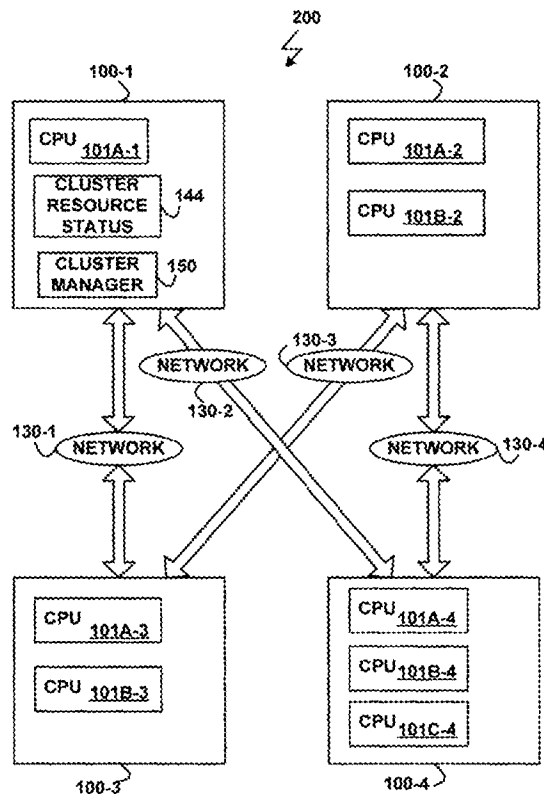


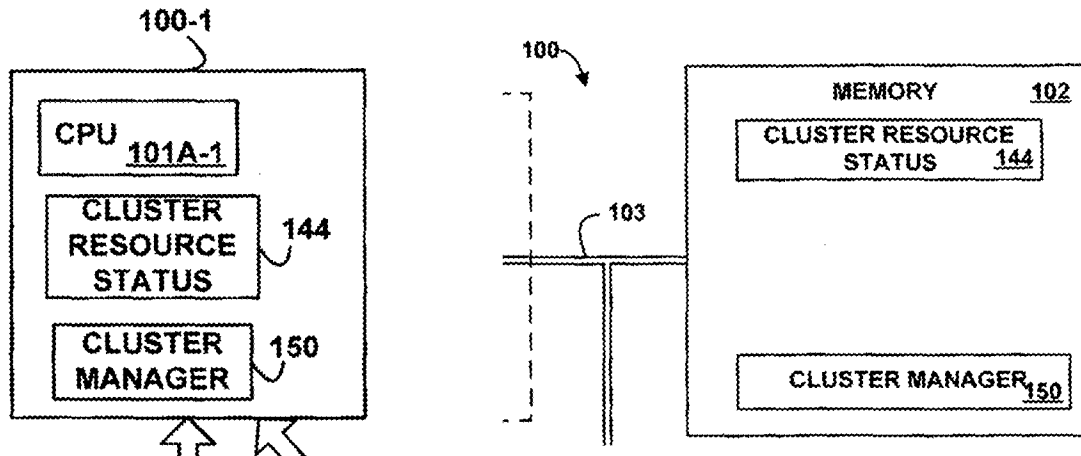
FIG. 2

Id., Fig. 2. Each computer system 100-1-100-4 is “referred to generically” as computer system 100 throughout *Bauer. Id.*, [0037].

[1.2a]: “storing license information for a computing cluster in a memory module associated with a computing node”

Bauer discloses or renders obvious limitation [1.2a]. Ex. C, ¶ 158. *Bauer* discloses storing license information for cluster 200 in cluster resource status 144 in memory 102 (the “memory

module”) of computer system 100-1 (the “computing node”). PA-2, [0016]-[0018], Fig. 1 (depicting cluster resource status 144 and cluster manager 150 in memory 102 of computer system 100-1), Fig. 2 (depicting cluster resource status 144 and cluster manager 150 in computer system 100-1):



Id., Fig. 1 (excerpted, right), Fig. 2 (excerpted, left). Cluster resource status 144 in memory 102 (“memory module”) may include a number of licenses field 340. Storing data in licenses field 340 teaches the claimed “storing license information for a computing cluster,” for it indicates “a maximum number of licensed resources available to the cluster 200” or “separate numbers of licenses for different types of resources” available to the cluster. *Id.*, [0043]; Ex. C, ¶ 158.

Further, memory 102 is associated with the computer system 100-1 and its CPUs 101A-1, as claimed. *See, e.g.*, PA-2, [0019] (“The memory bus 103 provides a data communication path for transferring data among the processors 101, the main memory 102, and the I/O bus interface unit 105.”), [0014] (“processor 101 executes instructions stored in the main memory 102 and may include one or more levels of on-board cache”), [0015] (“Memory may further be distributed and associated with different CPUs or sets of CPUs”). Figure 1, moreover, depicts cluster resource status 144 as being in memory 102 (claimed “memory module”):

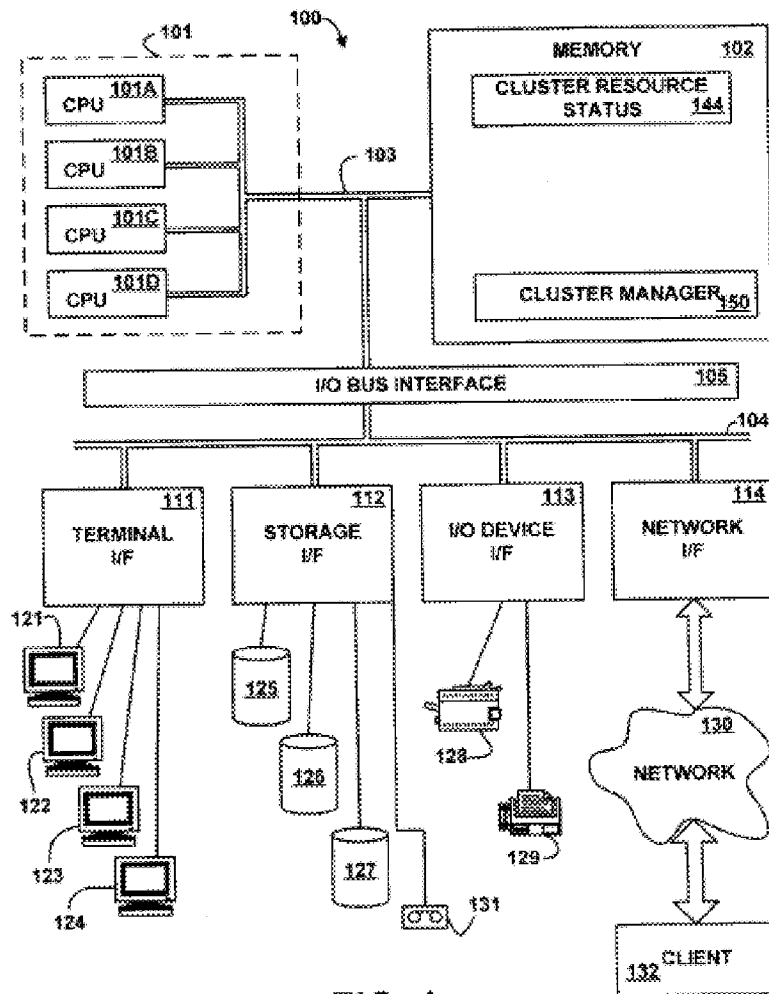


FIG. 1

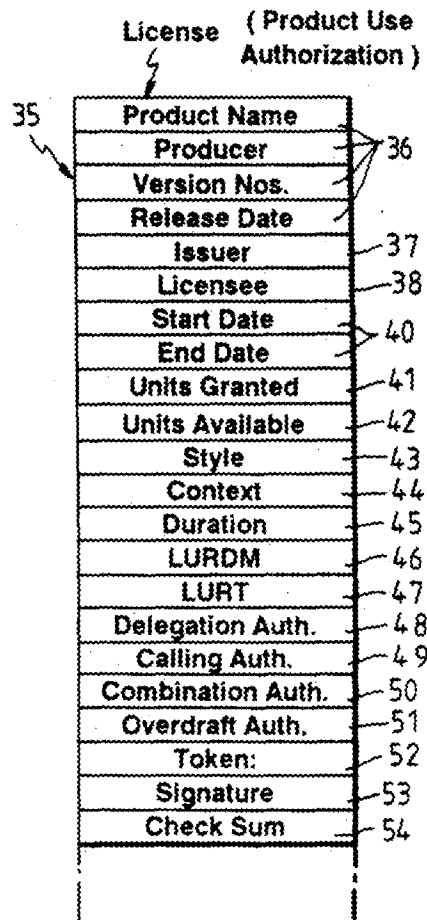
Id., Fig. 1. Memory 102 is a “random-access semiconductor memory for storing data and programs,” which includes both cluster resource status 144 and cluster manager 150. *Id.*, [0015].

[1.2b]: “wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster”

Bauer’s discloses or renders obvious limitation [1.2b]. Ex. C, ¶ 160. *Bauer* discloses receiving a “license to a number of resources in the cluster 200” and saving “the number of licensed resources in the number of licenses 340 in the cluster resource status 144.” PA-2, [0044]-[0045].

Bauer discloses the “number of licenses field 340 indicates the maximum number of licensed resources available to the cluster 200.” *Id.*, [0043]. These licensed resources may include, for example, CPUs or secondary storage. *See id.*, [0019], [0039]. In an embodiment in which each computing system includes a single licensable CPU (*id.*, [0014], [0017]), the number of licenses field 340 “identifies a size of the cluster” as required by [1.2b], since the number of licenses resources identifies the number of computing systems including active CPUs in the cluster. Ex. C, ¶ 160.

To the extent that *Bauer*’s number of licenses 340 does not teach or suggest a size of the cluster as required by [1.2b], including in embodiments with more than a single CPU per computer, [1.2b] would have nevertheless been obvious over *Wyman*. *Id.* *Wyman* discloses a license document that identifies the licensed product (36) and the number of granted license units (41):



PA-3, Fig. 2; *see also id.*, [12:12-34]. The license information can specify a maximum number of concurrent uses of a system. *Id.*, [20:20-27]. Wyman explains that a “concurrent use” license can be created by setting context 44 to a process ID, duration 45 to “transaction,” and style 43 to “allocative.” *Id.*, [20:20-27]. This license type allows multiple simultaneous uses of the licensed software, based on number of available license units. *Id.* The license can further specify, using context 44, that license units are granted on a per-node basis. *Id.*, [25:35-45].

As discussed, it would have been obvious to an ordinary artisan to modify *Bauer* according to the teachings of Wyman. *Supra* Section VI.B.3. This obvious modification would result in the incorporation of Wyman’s contextual allocation into *Bauer*’s numeric license check, which would permit greater licensing flexibility. *See* Ex. C, ¶ 162. Implementing the system from Wyman by

setting context 44 to a process ID, duration 45 to “transaction,” and style 43 to “allocative” would allow for the flexibility of *Wyman* while retaining the concurrent usage and per-node nature of the system of *Bauer*. Ex. C, ¶ 162. An ordinary artisan would understand how to substitute *Bauer*’s number of licenses field 340 for a more comprehensive set of parameters, such as those in *Wyman*, and have a reasonable expectation of success in doing so, particularly given the express examples provided in *Wyman*. Ex. C, ¶ 162; *see also supra* Section VI.B.3.

[1.2c]: “and at least one additional bundle-type characteristic of the cluster,”

Bauer and *Wyman* disclose or render obvious limitation [1.2c]. Ex. C, ¶ 163. In particular, additional fields in *Wyman*’s license document teaches and/or renders obvious [1.2c]. *Id.*

The ’416 patent describes a licensing system with system (bundle) and component (node)-level licensing parameters. *E.g.*, Ex. A, 5:40-50. At the system level, clusters are also categorized into types by additional parameters beyond size range, including such generic parameters as a “class” of the cluster. *E.g.*, *id.*, 8:1-5.

Similarly, *Wyman* discloses a license document with system-level licensing parameters, such as version number, and “essential” license terms, such as license duration and style of license allocation. PA-3, [11:55-12:60]. The license duration includes a start and end date, specified as an exact time when the license becomes value and ends. *Id.*, [12:19-22]. As an additional example, *Wyman* discloses a license document that includes a token field to store detailed feature authorizations, such as lists of authorized or denied features in a complex product. *Id.*, [33:17-24].

As discussed, it would have been obvious to an ordinary artisan to modify *Bauer*’s numeric license check with the contextual allocation system of *Wyman* to permit greater licensing flexibility. Ex. C, ¶ 166. An ordinary artisan would have understood how to substitute *Bauer*’s number of licenses field 340 for a more comprehensive set of parameters, such as those in *Wyman*,

and would have had a reasonable expectation of success doing so, particularly given the express examples provided in *Wyman. Id.*, *see also supra* Section VI.B.3.

[1.2d]: “and one or more node license parameters that identifies a characteristic of a computing node;”

Bauer and *Wyman* disclose or render obvious limitation [1.2d]. Ex. C, ¶ 167. In particular, additional fields in *Wyman*’s license document teaches or renders obvious limitation [1.2d]. *Id.*

The ’416 patent describes a licensing system with system (bundle) and component (node)-level licensing parameters. *E.g.*, Ex. A, 5:40-50. At the component level, the ’416 patent provides examples of the kind of information that can be “node license parameters that identif[y] a characteristic of a computing node,” including such generic parameters as a “class” of the node. *Id.*, 8:1-5.

Wyman discloses a license document that includes component-level licensing parameters, such as context 44, which specifies the contexts in which a license can be used. PA-3, [16:18-35]. This is shown below in Figure 3:

43	44	45	46
Style	Context	Duration	LURDM
Allocative	Network	Transaction	Constant
Consumptive	Execution Domain	Assignment	Table Lookup
Private	Login Domain	Immediate	Private
	Node ID		
	Process Family		
	Process		
	User Name		
	Product Name		
	Operating System		
	Platform ID		
	Private		

FIG. 3

Id., Fig. 3. License data is allocated or used within a specific context, and these contexts include a “specific network,” “a node,” “an operating system,” or “a specific hardware platform.” *Id.*, [16:24-28]. These contexts identify characteristics of computing nodes as required by [1.2d]. Ex.

C, ¶ 169. An ordinary artisan would further understand a Node_ID (as in Fig. 3) is a unique identifier for the node. *Id.*,

License 35 also defines a LURDM 46 or “license unit requirement determination method”. PA-3., Fig. 3. A LURDM stores information usable for calculating the number of units that should be allocated in response to a license request. *Id.*, [17:36-40]. One example is “table lookup,” and an example License Unit Requirement Table (LURT) for such embodiments is depicted in Figure 4:

License Unit Requirements Table			
Row Selector	Columns		
Platform ID	A	B	C
PC-0	10	230	-1
PC-1	12	230	-1
VAX 6210	158	300	150

FIG. 4

Id., Fig. 4. An LURT can specify if use is authorized in a particular licensing context. *Id.*, [17:36-18:59]. For example, an LURT can specify that a product is not authorized for use with a particular operating system. *Id.*, [19:27-60].

As discussed, it would have been obvious to an ordinary artisan to modify *Bauer*’s numeric license check with the contextual allocation system of *Wyman* to permit greater licensing flexibility. Ex. C, ¶ 171. As an example, *Wyman* teaches using LURDM and LURT components of the licensing document to prevent use by computing systems with potentially incompatible operating systems. PA-3, [19:27-60]. An ordinary artisan would have understood how to substitute *Bauer*’s number of licenses field 340 for a more comprehensive set of parameters, such as those

in *Wyman*, and would have had a reasonable expectation of success doing so, particularly given the express examples provided in *Wyman*. Ex. C, ¶ 171, *see also supra* Section VI.B.3.

[1.3]: “initializing the computing cluster in a first computing node;”

A cluster, being comprised of nodes, cannot function unless the cluster is initialized in at least one node. Ex. C, ¶ 172. *Bauer* discloses or teaches limitation [1.3] by saving “the number of licensed resources in the number of licenses 340 in the cluster resource status 144.” PA-2, [0044]-[0045].

CLUSTER RESOURCE STATUS		
COMPUTER ID	ACTIVE RESOURCES	INACTIVE RESOURCES
COMPUTER A	CPU A	CPU B, C, D
COMPUTER B	CPU A, B	CPU C, D
COMPUTER C	CPU A, B	CPU C, D
COMPUTER D	CPU A, B, C	CPU D

NUMBER OF LICENSES

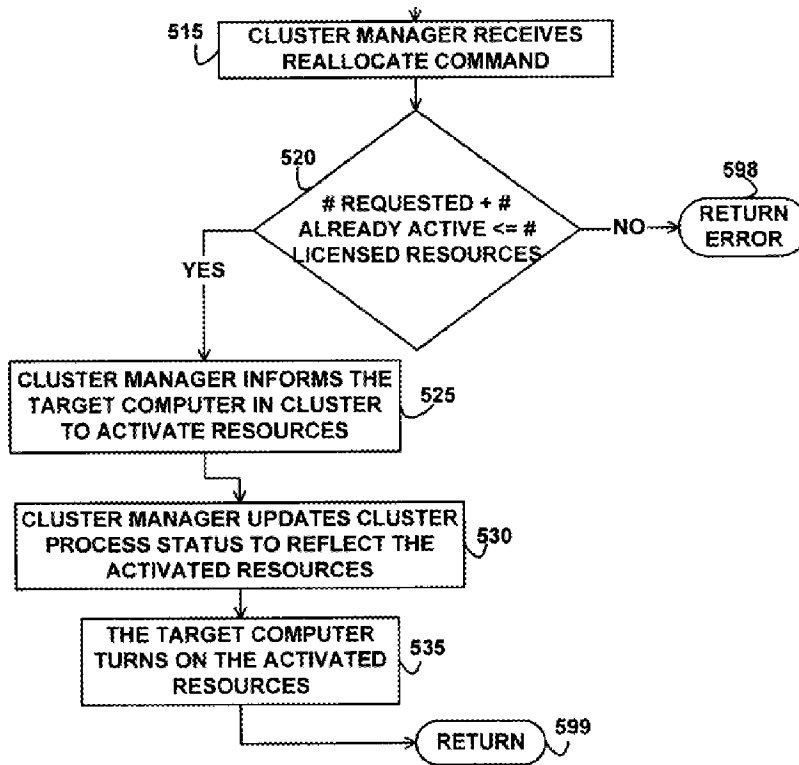
FIG. 3

PA-2, FIG. 3. After saving the number of licensed resources, cluster manager 150 then “activates [those] licensed resources,” which makes them “capable of being used” by systems in the cluster. [0045]. This is consistent with how the ’416 patent describes initializing in the specification. Ex. A, 7:33-41 (“[I]f at operation 520 the node has a node license that matches one or more required parameters, then control passes to operation 530 and the cluster may be initialized. In one

embodiment, the cluster may be initialized with as few as one member, i.e., the node that received the management command to create the cluster. The Right-to-Use License and the node license pool may be transferred to the newly created single node cluster, a node count parameter may be initialized (i.e., set to 1).”). Therefore, *Bauer* discloses or teaches initializing the computing cluster in a first computing node. Ex. C, ¶ 172.

[1.4]: “adding one or more available computing nodes to the computing cluster;”

A cluster, being comprised of nodes, necessarily requires that nodes be added to it. Ex. C, ¶ 173. In addition to activating resources upon receipt of a license, *Bauer* discloses activating resources in response to a failure of a licensed resource. *Id.* *Bauer* discloses or teaches limitation [1.4] at least based on its teaching of activating resources in response to a failure of a licensed resource. In Fig. 5, cluster manager 150 determines whether activating the number of requested resources (in a target computer) would exceed the number of licenses 340 (in cluster resource status 144) (step 520). *See* PA-2, [0049].



PA-2, FIG. 5.

If activating the requested resources would not exceed that number, cluster manager 150 informs the target computer to activate each of the requested resources (step 525), while updating the cluster resource status 144 to account for the activated resources (step 530). PA-2, [0051]-[0053].

As discussed above, it would have been obvious to an ordinary artisan at the time of the alleged invention to modify *Bauer*'s numeric license check with the contextual allocation system of *Wyman* to permit greater licensing flexibility. *See supra* Section VI.B.3; Ex. C, ¶ 175. In addition to supporting system-level licensing parameters such as number of concurrent active nodes, license duration, and style of license allocation, the contextual allocation system of *Wyman* permits component-level licensing parameters such as permitting or forbidding nodes based on computing system compatibility. Ex. C, ¶ 175.

The modified system of *Bauer* could check whether activation of a CPU on a target computing system would comply with a licensing document as described in *Wyman*, including both system-level requirements (i.e., number of concurrent active nodes, license duration, style of license allocation) and component-level requirements (i.e., computing system compatibility requirements). Ex. C, ¶ 176.

An ordinary artisan at the time of the alleged invention would have appreciated that the number of concurrent active nodes can comply with a licensing document as described in *Wyman*, when the number of such nodes is less than a maximum number of such nodes. Ex. C, ¶ 177.

[1.5]: “and activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”

For reasons similar to those explained above with respect to limitations [1.2b], [1.3], and [1.4], *Bauer* and/or *Wyman* teaches or suggests limitation [1.5].

As explained above with respect to Fig. 5, *Bauer* discloses providing instructions to activate resources in a cluster when the resulting number of activated resources complies with a maximum number of licenses specified in the received license. PA-2, [0045] (“cluster manager 150 activates licensed resources at any computer or computers in the cluster 200, *where the number of activated resources is less than or equal to the number of licensed resources*”); [0051] (“If the determination at block 520 is true, then number of requested resources plus the number of already active resources in the cluster 200 is less than or equal to the number of licensed resources 340, so control continues to block 525 where the cluster manager 150 instructs the target computer system 100 specified by the reallocate command to activate the specified resource or resources.”).

Bauer additionally discloses separately providing instructions to a target computer system to turn on the previously activated resources. PA-2, [0052] (“Control then continues to block 535

where the cluster manager 150 sends an activation request to the target computer system 100, which in response turns on or activates the resources, so that they are available for use.”).

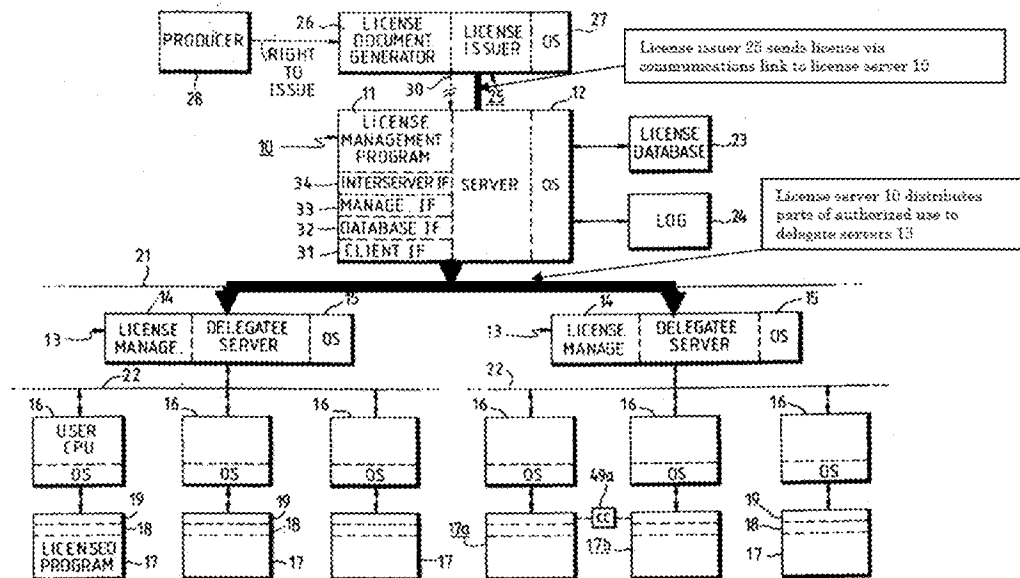
As described herein, activation of a computing cluster in the '416 patent amounts to proceeding with provision of cluster services. *Supra* Section VI(A)(2)[1.5]; Ex. C, ¶ 180. When *Bauer* recovers from a failure of all resources, according to the method of Fig. 5, *Bauer*'s cluster manager instructs a target computer system to activate resources (block 525) and then sends an activation request to the target computer system, which in response turns on or activates the resources, so that they are available for use (block 535). Therefore, a skilled artisan would have found it obvious that *Bauer*'s cluster manager activates *Bauer*'s computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter. PA-2, 8:19-20. Ex. C, ¶ 180.

As discussed above, it would have been obvious to an ordinary artisan at the time of the alleged invention to modify *Bauer*'s numeric license check with the contextual allocation system of *Wyman* to permit greater licensing flexibility. As described herein, the modified system of *Bauer* can check whether activation of a CPU on a target computing system would comply with a licensing document as described in *Wyman*, when determining whether to activate the CPU on the target computing system. Ex. C, ¶ 181. As that check has been performed and necessarily passed, the modified system of *Bauer* necessarily activates the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter. Ex. C, ¶ 181.

5. **Claim 2: “The method of claim 1, wherein storing license information for a computing cluster in a memory module associated with a computing node comprises retrieving licensing information from an external licensing server.”**

Bauer in view of *Wyman* teaches claim 2. Ex. C, ¶ 182. *Bauer* teaches receiving a license “via the network 130,” which may originate from a “manufacturer of the computer system 100.” [0044].

To the extent that *Bauer*’s receipt of a license via a network does not teach “retrieving licensing information from an external licensing server” as required by claim 2, *Wyman* teaches it. Ex. C, ¶ 183. For example, Fig. 1 depicts a license management facility with a license issuer 25, a license server 10, and delegate servers 13. License server 10 has “no authority to originate a license;” it must instead receive licenses via a “network” “communications link” from the license issuer 25 (claimed “retrieving licensing information from an external licensing server”). PA-3, 10:37-50. License server 10 stores the license and may distribute parts of the authorized use to delegate servers 13. PA-3, [11:12-16]; Fig. 1 (annotated):



As discussed above, it would have been obvious to an ordinary artisan at the time of the alleged invention to modify *Bauer*'s system for receiving a license over a network with the system of *Wyman* which requires a license issuer to send a license to permit centralized distribution of licenses. Ex. C, ¶ 184. *Bauer* already includes a network 130 for receiving a license ([0044]); *Wyman*'s system for distributing licenses allows a producer to grant the "right to issue" to a particular license issuer. PA-3, [10:41-45]. An ordinary artisan at the time of the alleged invention would understand that this arrangement would provide for increased security of licenses and better control for the producer of licensed software or other resources. Ex. C, ¶ 184.

6. **Claim 3: "The method of claim 1, wherein the bundle-type parameter identifies at least one of a minimum or maximum node count threshold for the computing cluster."**

Bauer in view of *Wyman* teaches claim 3. Ex. C, ¶ 185. As explained above with respect to at least limitation [1.2b], *Bauer* discloses receiving a "license to a number of resources in the cluster 200" and saving "the number of licensed resources in the number of licenses 340 in the cluster resource status 144," to indicate "the maximum number of licensed resources available to the cluster 200." [0043]-[0045]. However, to the extent that *Bauer* does not disclose that number of licenses field 340 includes a "bundle-type parameter" that "identifies at least one of a minimum or maximum node count threshold for the computing cluster," *Wyman* does. Ex. C, ¶ 185.

As explained above, *Wyman* discloses a license document that identifies, *inter alia*, the licensed product (36) and the number of granted license units (41). PA-3, [12:12-37]. The license can further specify using context 44 that license units are granted on a per-node basis. *Id.*, [25:35-45]. Because the license units can be granted on a per-node basis, the number of granted license units acts as a "maximum node count threshold for the computing cluster" as required by claim 3. Ex. C, ¶ 186.

Bauer's existing extensibility for license data would be complemented by *Wyman*'s more extensible license that includes the number of license units, among other fields such as start/end date, context, and duration. An ordinary artisan would understand how to substitute *Bauer*'s number of licenses field 340 for a more comprehensive set of parameters as in *Wyman*, providing for additional functionality. Ex. C, ¶ 187. *Wyman* even calls its licenses an "improvement" to "cluster licenses," such as those used in *Bauer*, because it "provides a general mechanism which allows an effectively unlimited range of allocation scopes to be defined." PA-3, [25:44-53]; Ex. C, ¶ 187.

7. **Claim 4: "The method of claim 1, wherein: the node license parameters identify one or more bundle-types compatible with the computing node."**

Bauer in view of *Wyman* teaches claim 4. As explained above with respect to at least limitations [1.2b] and [1.2d], *Bauer* discloses receiving a "license to a number of resources in the cluster 200" and saving "the number of licensed resources in the number of licenses 340 in the cluster resource status 144," to indicate "the maximum number of licensed resources available to the cluster 200." [0043]-[0045]. However, to the extent that *Bauer* does not identify one or more bundle-types "compatible with the computing node" *Wyman* does. Ex. C, ¶ 188.

For example, *Wyman* discloses a license document that includes a license units requirement table (LURT). [17:36-43]; Fig. 4 (depicting example LURT) As described in *Wyman*, a licensing context can specify a "network, ... a node, ... an operating system, [or] a specific hardware platform." 16:18-35. The LURT can include rows with corresponding entries that indicate that a product is unsupported by (and therefore unauthorized for use with) a particular operating system. PA-3, [19:27-60]; Ex. C, ¶ 189.

As discussed above, it would have been obvious to an ordinary artisan at the time of the alleged invention to modify *Bauer*'s numeric license check with the contextual allocation system of *Wyman* to permit greater licensing flexibility. As an example of such flexibility, *Wyman* teaches using LURDM and LURT components of the licensing document to prevent use by computing systems with potentially incompatible operating systems. PA-3, [18:37-21:6]; Ex. C, ¶ 190.

8. **Claim 5: “The method of claim 1, wherein adding one or more available computing nodes to the computing cluster comprises assigning a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.”**

Bauer in view of *Wyman* teaches claim 5. Ex. C, ¶ 191. For the reasons discussed above with respect to limitations [1.4], *Bauer* in view of *Wyman* teaches “adding one or more available computing nodes to the computing cluster”. *Wyman* further teaches that the addition of such computing nodes can include “assigning a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.” Ex. C, ¶ 191.

Wyman discloses a computing cluster including a licensing server that administers licenses for nodes. PA-3, [6:49-67]. *Wyman* discloses an embodiment having a license that includes cluster licensing parameters (“brand-type”), such as temporal conditions on license use. PA-3, [18:38-62]. *Wyman* therefore teaches “a brand-type associated with the computing cluster.” Ex. C, ¶ 192.

Wyman discloses authorizing product use by granting license unit allocations. PA-3, [21:27-42]. An authorization request includes a collection of context information, which is filtered to generate a license context, such as a node. PA-3, [24:18-49]. *Wyman* therefore teaches “the computing node license” because *Wyman* discloses granting license unit allocations to authorization requests associated with a node. Ex. C, ¶ 193.

Wyman discloses that authorization requests that filter to the same license contexts can share allocated license units. PA-3, [24:50-25:12]. However, authorization requests that filter to the same license context still require verification. *Id.*, [25:1-5]. When a temporal condition on license use is unsatisfied, the verification will fail and the authorization request will be denied. *Id.*, [23:48-53]. *Wyman* therefore teaches “when the computing node license is incompatible with a brand-type” because an authorization request from a licensed node can fail due to incompatibility with system-level temporal restrictions. Ex. C, ¶ 194.

Wyman discloses that an authorization request can be granted under a second license with different system-level parameters when the request cannot be granted under the first license. PA-3, [13:40-50]. *Wyman* therefore teaches “assigning a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster” because a new license unit allocation can be granted under a second license when the authorization request from the licensed node failed due to incompatibility with system-level temporal restrictions of the first license, a license administered for client nodes of a distributed computer system. Ex. C, ¶ 195.

9. **Claim 6 “The method of claim 1, further comprising initiating a license check while the computing system is operating, wherein the license check evaluates at least one of:**
 - a license status of the computing node;**
 - a node count in the computing cluster; and**
 - a brand-type associated with the computing node.”**

Bauer in view of *Wyman* teaches claim 6. Ex. C, ¶ 196. *Wyman* teaches a user node 16 making a request allocation call to a license server during execution of a program on the client. *See generally* PA-3, [21:26-45]; [23:3-25:43]; Fig. 5.

The request allocation call includes a collection of the “full context” for a request allocation call, including information the user node. PA-3, [24:18-24:25]. The call to the Lm_Request_Allocation() function (step 61, Fig. 5) is a “request that license units be allocated to the current context.” PA-3, [21:46-48]. In response to the request allocation call, the license manager will “look in its list of available product use authorizations” to determine a match to the product identifier in that call. *Id.*, [24:26-31]. If a match is found, the license manager can retrieve the appropriate context template and use the context information and the context template to determine whether a license has been granted for that use (the claimed “evaluates ... a license status of the computing node”). *Id.*, [24:31:68]. For example, when the context template specifies that only node-ID is considered, a license is granted once per user node. *Id.*, [25:35-45].

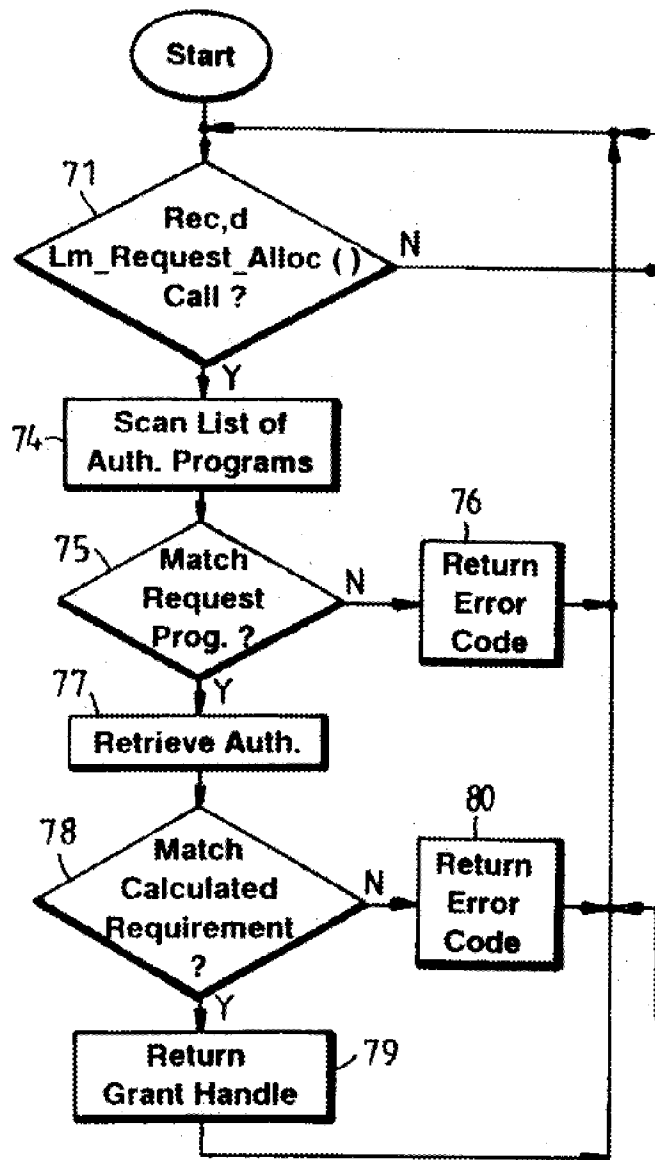
Wyman also teaches that request allocation calls can be made “at a number of points in executing a program, rather than only on start-up,” such as to determine licensure of a particular feature (the claimed “initiating a license check while the computing system is operating”). PA-3, [21:32-36]; *see also id.*, [23:20-23].

Bauer’s existing extensibility for license data would be complemented by *Wyman*’s more extensible license. Ex. C, ¶ 199. For example, *Wyman*’s more extensible license can enable a request allocation to be made when making use of a feature in a licensed program without requiring each in-use copy of the program to have a respective license to that feature at all times. PA-3, [2:35-52]; [14:62-15-17]; [21:32-36]. An ordinary artisan would understand how to substitute *Bauer*’s number of licenses field 340 for a more comprehensive set of parameters as in *Wyman*, providing for additional functionality. Ex. C, ¶ 199. *Wyman* even calls its licenses an “improvement” to “cluster licenses,” such as those used in *Bauer*, because it “provides a general mechanism which allows an effectively unlimited range of allocation scopes to be defined.” PA-

3, [25:44-53]. Indeed, enabling a user to request allocation of a feature during the execution of a program provides additional functionality, including the ability to license out features to a large group of users on an as-needed basis. Ex. C, ¶ 199.

10. **Claim 7: “The method of claim 1, wherein the license check comprises implementing an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”**

Bauer in view of *Wyman* teaches claim 7. Ex. C, ¶ 200. As discussed above with respect to claim 6, *Wyman* discloses a user node making a request allocation call to a license server during execution of a program on the client. *Supra* Section VI.B.9. The license server can determine whether the allocation request can be granted based on a row of the LURT containing a “negative one (-1) to indicate that [an] operating system was unsupported.” PA-3, [19:50-60]; *see also id.*, [19:27-50]. If a product is unsupported, a license allocation request cannot be granted (the claimed “incompatibility between the computing node and the computing cluster”). *See* PA-3, 19:50-60 (explaining how to indicate an unsupported operating system). If a grant cannot be made in response to a request allocation call, the license server would return an error code (the claimed “error routine that inhibits operation of the computing node”). PA-3, [23:45-55]; *see also* Fig. 5 (block 80: “Return Error Code”).



Bauer's existing extensibility for license data would be complemented by *Wyman*'s more extensible license that enables the determination of an incompatibility. Ex. C, ¶ 201. An ordinary artisan would understand how to substitute *Bauer*'s number of licenses field 340 for a more comprehensive set of parameters as in *Wyman*, providing for additional functionality. Ex. C, ¶ 201. *Wyman* even calls its licenses an "improvement" to "cluster licenses," such as those used in *Bauer*, because it "provides a general mechanism which allows an effectively unlimited range of

allocation scopes to be defined.” PA-3, [25:44-53]. Indeed, enabling the licensing in *Bauer* to account for incompatibility and follow an error routine, as in *Wyman*, would create a more functional system that does not risk system errors due to incompatibility. Ex. C, ¶ 201.

11. Claim 8: “The method of claim 1, further comprising adding a computing node to the activated computing cluster.”

Bauer discloses adding a computing node to an activated computing cluster for reasons similar to those discussed above with respect to limitation [1.4]; Ex. C, ¶ 202. *Bauer* discloses the cluster manager of the computer system providing instructions to activate resources in the computing cluster and update the cluster resource status to reflect the activation of the resources. PA-2, [0045], [0051]-[0052].

12. Independent Claim 9

[9.1] “A computing system comprising at least two computing nodes, wherein at least one of the computing nodes comprises:”

Bauer discloses limitation [9.1] for reasons similar to those explained above with respect to limitations [1.1] and [1.2a]. Ex. C, ¶ 203.

[9.2a] “a cluster module stored on a non-transitory computer readable medium,”

Bauer discloses limitation [9.2a]. Ex. C, ¶ 204. The ’416 patent states that the “modules” in the node depicted in Fig. 4 are just logic that may be embodied in software. Ex. A, 6:4-13; *see also* Section VI.A.9[9.2a]. The node includes, among other modules, a cluster module 410. PA-2, Fig. 4. As shown below, the computer systems (100) of *Bauer* include software stored on a non-transitory computer readable medium, i.e., “a logic module” that manages “cluster-level compatibility functions” and thus corresponds to the claimed “cluster module.” Ex. C, ¶ 204.

In particular, *Bauer*’s nodes include a cluster manager 150 that performs certain steps of limitations [9.2b], [9.2c], or [9.2d] (the claimed “cluster module”). PA-2, Fig. 1; Ex. C, ¶ 205. Like

the '416 patent, *Bauer* also describes that its various embodiments (such as the cluster manager 150) “may be implemented” as “modules” that are “resident” in storage devices in the depicted computer systems 100 (and/or 101-1) and cause processor 101 to execute steps (the claimed “module stored on a non-transitory computer readable medium”). PA-2, [0029], [0031]; Ex. C, ¶205. Similarly, *Wyman* teaches that license server 10 executes license management program 11, which an ordinary artisan in the art at the time of the alleged invention would understand to be software. 9:25-30; Ex. C, ¶ 205.

[9.2b]: “the cluster module configured to manage compatibility operations associated with a computing cluster,”

Bauer in view *Wyman* of discloses limitation [9.2b]. The cluster manager 150 of *Bauer* includes logic to manage the status of licensable resources via the cluster resource status (the claimed “cluster module”). PA-2 [0018]. *Wyman* discloses managing compatibility operations through license checks and system-level licensing parameters, such as version number, and “essential” license terms, such as license duration and style of license allocation (the claimed management of “compatibility operations associated with a computing cluster”). PA-3, [11:55-12:60]. The license duration includes a start and end date, specified as exact times when the license becomes valid and becomes invalid. *Id.*, [12:19-22]. *Wyman* also discloses a license document that includes a token field to store detailed feature authorizations, such as lists of authorized or denied features in a complex product. *Id.*, [33:17-24]. All of these parameters are used to ensure that the clusters are compatible with other network resources. Ex. C, ¶ 206. For example, problems would occur if a cluster’s license duration conflicted with other network resources, or if its authorized features were incompatible with those of other devices on the network. *Id.*

As discussed, it would have been obvious to an ordinary artisan to modify *Bauer*’s licensing management system with the contextual allocation system of *Wyman* to permit greater

licensing flexibility. Ex. C, ¶ 207. An ordinary artisan would have understood how to substitute *Bauer*'s number of licenses field 340 for a more comprehensive set of parameters, such as those in *Wyman*, and would have had a reasonable expectation of success doing so, particularly given the express examples provided in *Wyman*. *Id.*; *see also supra* Section VI.B.3.

[9.2c]: “the cluster module initializing a computing cluster in the first computing node,”

Bauer discloses or teaches limitation [9.2c], as this limitation is substantially similar to limitation [1.3]. *Supra* Section VI.B.4[1.3]; Ex. C, ¶ 208. While this limitation specifies that the method is performed by a “cluster module,” said “cluster module” is just software stored in the memory of the node. Ex. A 6:8-11. The nodes of *Bauer* likewise function through logic embodied in software stored in memory. *See, e.g.*, PA-2 [0018], [0029]. So *Bauer* discloses this limitation for the reasons discussed above in connection with limitation [1.3]. *Supra* Section VI.B.4[1.3]; Ex. C, ¶ 208.

[9.2d]: “the cluster module adding one or more available computing nodes to the computing cluster;”

Bauer discloses or teaches limitation [9.2d], as this limitation is substantially similar to limitation [1.4]. *Supra* Section VI.B.4[1.4]; Ex. C, ¶ 209. While this limitation specifies that the method is performed by a “cluster module,” said “cluster module” is just software stored in the memory of the node. Ex. A 6:8-11. The nodes of *Bauer* likewise function through logic embodied in software stored in memory. *See, e.g.*, PA-2 [0018], [0029]. So *Bauer* discloses this limitation for the reasons discussed above in connection with limitation [1.4]. *Supra* Section VI.B.4[1.4]; Ex. C, ¶ 209.

[9.3a] “a node module stored on a non-transitory computer readable medium,”

For reasons, including those discussed above with respect to [9.2a], *Bauer* teaches [9.3a]. *Supra* Section VI.B.12[9.2a]. As further described below, *Bauer* teaches logic embodied in software, a cluster manager 150, that performs certain steps of limitations [9.3b], [9.3c], [9.3d], and [9.3e] (the claimed “node module”). Ex. C, ¶ 210. Similarly, *Wyman* teaches that license server 10 executes license management program 11, which an ordinary artisan in the art at the time of the alleged invention would understand to be software (the claimed “node module”). PA-3, 9:25-30; EX. C, ¶ 210.

[9.3b] “the node module configured to manage compatibility operations associated with the at least two computer nodes,”

For reasons including those discussed above with respect to limitation [9.2a], *Bauer* in view of *Wyman* teaches [9.2b]. Ex. C, ¶ 211. *Bauer* discloses that logic embodied in software in the cluster manager manages the status of licensable resources using the cluster resource status. PA-2, [0018].

Wyman discloses a license server configured to administer licenses for a distributed computing system including multiple client nodes. PA-3, [6:49-67]. The licenses include node-level licensing parameters, such as context 44, which specifies the contexts in which a license can be used. PA-3, [16:18-35]. This is shown below in Figure 3:

43 Style	44 Context	45 Duration	46 LURDM
Allocative	Network	Transaction	Constant
Consumptive	Execution_Domain	Assignment	Table Lookup
Private	Login_Domain	Immediate	Private
	Node_ID		
	Process_Family		
	Process		
	User_Name		
	Product_Name		
	Operating_System		
	Platform_ID		
	Private		

FIG. 3

Id., Fig. 3. License data is allocated or used within a specific context, and these contexts include a “specific network,” “a node,” “an operating system,” or “a specific hardware platform.” *Id.*, [16:24-28]. These contexts identify characteristics of computing nodes as required by limitation [1.2d]. Ex. C, ¶ 212. An ordinary artisan would further understand a Node_ID (as in Fig. 3) is a unique identifier for the node. *Id.*

License 35 also defines a LURDM or “license unit requirement determination method,” 46. PA-3, Fig. 3. A LURDM stores information usable for calculating the number of units that should be allocated in response to a license request. *Id.*, [17:36-40]. One example is “table lookup,” and an example License Unit Requirement Table (LURT) for such embodiments is depicted in Figure 4:

License Unit Requirements Table			
Row Selector	Columns		
Platform ID	A	B	C
PC-0	10	230	-1
PC-1	12	230	-1
VAX 6210	158	300	150

FIG. 4

Id., Fig. 4. An LURT can specify if use is authorized in a particular licensing context. *Id.*, [17:36-18:59]. For example, an LURT can specify that a product is not authorized for use with a particular operating system. *Id.*, [19:27-60].

As discussed, it would have been obvious to an ordinary artisan to modify *Bauer*'s cluster manager to act as the license server of *Wyman*. The modified cluster manager could implement the contextual allocation system of *Wyman* to permit greater licensing flexibility. Ex. C, ¶ 214. As an example, *Wyman* teaches using LURDM and LURT components of the licensing document to prevent use by computing systems with potentially incompatible operating systems. PA-3, [18:37-21:6]. An ordinary artisan would have understood how to substitute the license server of *Wyman* for the cluster manager of *Bauer* and had a reasonable expectation of success in doing so, given the simple functionality of the cluster manager of *Bauer* and the extensive and detailed disclosure of *Wyman*. Ex. C, ¶ 214; *see also supra* Section VI.B.3.

[9.3c] “the node module storing license information for the computing cluster,”

Bauer discloses limitation [9.3c], as this limitation is substantially similar to limitation [1.2a]. *Supra* Section VI.B.4[1.2a]; Ex. C, ¶ 215. This limitation specifies that the method is performed by a “node module,” which is just software stored in the memory of the node. Ex. A

6:8-11. The nodes of *Bauer* likewise function through logic embodied in software stored in memory. *See, e.g.*, PA-2 [0018], [0029]. So *Bauer* discloses this limitation for the reasons discussed above in connection with limitation [1.2a]. *Supra* Section VI.B.4[1.2a]; Ex. C, ¶ 215.

[9.3d] “wherein the licensing information includes a bundle-type parameter that identifies a size of the computing cluster and at least one additional bundle-type characteristic of the computing cluster,”

This limitation is substantially similar to limitations [1.2b] and [1.2c]. *Supra* Section VI.B.4[1.2b]-[1.2c]; Ex. C, ¶ 216. This limitation specifies that the method is performed by a “node module,” which is just software stored in the memory of the node. Ex. A 6:8-11. The nodes of *Bauer* likewise function through logic embodied in software stored in memory. *See, e.g.*, PA-2 [0018], [0029]. So *Bauer* discloses this limitation for the reasons discussed above in connection with limitations [1.2b] and [1.2c]. *Supra* Section VI.B.4[1.2b]-[1.2c]; Ex. C, ¶ 216.

[9.3e] “and one or more node license parameters that identifies a characteristic of a first computing node;”

Bauer discloses [9.3e], as this limitation is substantially similar to limitation [1.2d]. *Supra* Section VI.B.4[1.2d]; Ex. C, ¶ 217. This limitation specifies that the method is performed by a “node module,” which is just software stored in the memory of the node. Ex. A 6:8-11. The nodes of *Bauer* likewise function through logic embodied in software stored in memory. *See, e.g.*, PA-2 [0018], [0029]. So *Bauer* discloses this limitation for the reasons discussed above in connection with limitation [1.2d]. *Supra* Section VI.B.4[1.2d]; Ex. C, ¶ 217.

[9.4a] “and a management module stored on a non-transitory computer readable medium, the management module configured to manage compatibility operations with one or more external servers,”

Bauer and *Wyman* disclose limitation [9.4a]. The ’416 patent discloses a “management module 450 may include an external management control module 452 and an external license server access module 454 that provide an interface with an external license server 460.” Ex. A,

6:64-67. *Bauer* describes the cluster manager receiving a “license via the network . . . from a manufacturer of the computer system 100 or from a licensor of the associated resources.” PA-2, [0044]. And *Wyman* discloses receiving a license over a communications link from a “distributor who has received the authority to grant licenses from the producer or owner.” PA-3, [10:30-57]. Both of these functions require logic embodied in software “to manage compatibility operations with one or more external servers.” Ex. C, ¶ 218. Otherwise, *Bauer’s* computer system would be unable to communicate with and receive licensing information from an external server. Ex. C, ¶ 218.

[9.4b] “the management module activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”

Bauer discloses limitation [9.4b], as this limitation is substantially similar to limitation [1.5]. *Supra* Section VI.B.4[1.5]; Ex. C, ¶ 219. This limitation specifies that the method is performed by a “management module,” which is just software stored in the memory of the node. Ex. A 6:8-11. The nodes of *Bauer* likewise function through logic embodied in software stored in memory. *See, e.g.*, PA-2 [0018], [0029]. So *Bauer* discloses this limitation for the reasons discussed above in connection with limitation [1.5]. *Supra* Section VI.B.4[1.5]; Ex. C, ¶ 219.

13. **Claim 10: “The computing system of claim 9, wherein the cluster module comprises a create cluster module to aggregate the at least two computing nodes into a clustered computing system.”**

Bauer discloses claim 10 for reasons similar to those discussed above for limitation [9.2a] (“cluster module comprises a create cluster module”). *Bauer* further discloses that the cluster manager receives a license to a number of resources in the cluster and saves the number of licensed resources in the number of licenses 340 in the cluster resource status 144. PA-2, [0044], [0045]. *Bauer* discloses that the cluster manager of the computer system provides instructions to activate

resources in the computing cluster and update the cluster resource status to reflect the activation of the resources (claimed “aggregate the at least two computing nodes into a clustered computing system”). PA-2, [0045], [0051]-[0052]; Ex. C, ¶ 220.

14. **Claim 11: “The computing system of claim 10, wherein the cluster module maintains a right to use license comprising licensing information for a computing cluster and a node license pool comprising licensing information for nodes in the computing cluster.”**

Bauer in view of *Wyman* teaches claim 11. Ex. C, ¶ 221. *Wyman* discloses a license document that includes identification of the licensed software product by product name, producer, version number, and “essential” license terms, such as license duration and style of license allocation. PA-3, [11:55-12:60]. The license document further specifies a number of license units (and available units) and licensing information governing the allocation of license units. *Id.*, [12:10-59].

Bauer in view of *Wyman* teaches a license document comprising licensing information for a computing cluster and comprising a node license pool comprising licensing information for nodes in the computing cluster. Ex. C, ¶ 221.

15. **Claim 12: “The computing system of claim 10, further comprising an add node module to add a computing node to the clustered computing system.”**

Bauer discloses claim 12. Ex. C, ¶ 223. This limitation is substantially similar to limitation [9.2a] (“further comprising an add node module”) and claim 8 (claimed “to add a computing node to the clustered computing system”). *Supra* Section VI.B.12[9.2a]; Ex. C, ¶ 223. This limitation specifies that the method is performed by an “add node module,” which is just software stored in the memory of the node. Ex. A 6:8-11. The nodes of *Bauer* likewise function through logic embodied in software stored in memory. *See, e.g.*, PA-2 [0018], [0029]. So *Bauer* discloses this

limitation for the reasons discussed above in connection with limitation [9.2a] and claim 8. *Supra* Section VI.B.12[9.2a]; *see also* PA-2, [0045], [0051]-[0052]; Ex. C, ¶¶ 202, 204.

16. Independent Claim 14

The only difference between claims 1 and 14 is that claim 14 recites a computer program product stored on a non-transitory computer-readable medium and executed by a computer processor to perform the method recited in claim 1. The nodes of *Bauer* have logic defined by a computer program stored in the nodes' memory and that executes on the nodes' processors (CPUs). PA-2, Figs. 1 and 2, [0029], [0037]-[0039]. Thus, the discussion of claim 1, and all its elements, is incorporated herein to the discussion of claim 14 and all its elements, in its entirety

[14.1] “A computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor,”

For reasons discussed above with respect to at least limitations [1.1], [9.1] and [9.2a], *Bauer* teaches limitation [14.1]. Ex. C, ¶ 225; *supra* Sections VI.B.4[1.1], VI.B.12[9.1]-[9.2a].

[14.2a]: “configure the processor to: store license information for a computing cluster in a memory module associated with a computing node,”

For reasons similar to those discussed above with respect to limitation [1.2a], *Bauer* discloses limitation [14.2a]. Ex. C, ¶ 226; *supra* Section VI.B.4[1.2a].

[14.2b]: “wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and”

For reasons similar to those discussed above with respect to limitation [1.2b], *Bauer* in view of *Wyman* teaches limitation [14.2b]. Ex. C, ¶ 227; *supra* Section VI.B.4[1.2b].

[14.2c]: “at least one additional bundle-type characteristic of the cluster, and”

For reasons similar to those discussed above with respect to limitation [1.2c], *Bauer* in view of *Wyman* discloses limitation [14.2c]. Ex. C, ¶ 228; *supra* Section VI.B.4[1.2c].

[14.2d] “one or more node license parameters that identifies a characteristic of a computing node;”

For reasons similar to those discussed above with respect to limitation [1.2d], *Bauer* in view of *Wyman* discloses limitation [14.2d]. Ex. C, ¶ 229; *supra* Section VI.B.4[1.2d].

[14.3] “initialize the computing cluster in a first computing node;”

For reasons similar to those discussed above with respect to limitation [1.3], *Bauer* in view of *Wyman* discloses limitation [14.3]. Ex. C, ¶ 230; *supra* Section VI.B.4[1.3].

[14.4] “add one or more available computing nodes to the computing cluster; and”

For reasons similar to those discussed above with respect to limitation [1.4], *Bauer* in view of *Wyman* discloses limitation [14.4]. Ex. C, ¶ 231; *supra* Section VI.B.4[1.4].

[14.5] “activate the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.”

For reasons similar to those discussed above with respect to limitations [1.2b], [1.3], [1.4], and [1.5], *Bauer* in view of *Wyman* discloses limitation [14.5]. Ex. C, ¶ 232; *supra* Section VI.B.4[1.1], [1.3]-[1.5].

17. Claim 15: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to retrieve licensing information from an external licensing server.”

As discussed for claim 14, *Bauer* and *Wyman* render obvious “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium

which, when executed by a computer processor, configure the processor” to perform various steps. *See supra* Section VI.B.16[14.1]-[14.2a]; Ex. C, ¶ 233. The rest of this claim limitation is substantially similar to claim 2. *See* Section VI.B.5; Ex. C, ¶ 233. *Bauer* and *Wyman* render obvious this limitation for the same reasons discussed in connection with claim 2. *Supra* Section VI.B.5; Ex. C, ¶233.

18. **Claim 16: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to assign a computing node a new node license when the computing node license is incompatible with a brand-type associated with the computing cluster.”**

As discussed for claim 14, *Bauer* and *Wyman* render obvious “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor, configure the processor” to perform various steps. *See supra* Section VI.B.16[14.1]-[14.2a]; Ex. C, ¶ 234. The rest of this limitation is substantially similar to claim 5. *See* Section VI.B.8; Ex. C, ¶ 234. As discussed above, *Bauer* and *Wyman* render obvious this limitation for the same reasons discussed in connection with claim 5. *Supra* Section VI.B.8; Ex. C, ¶ 234.

19. **Claim 17: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to initiate a license check while the computing system is operating, wherein the license check evaluates at least one of: a license status of the computing node; a node count in the computing cluster; and a brand-type associated with the computing node.”**

As discussed for claim 14, *Bauer* and *Wyman* render obvious “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor, configure the processor” to perform various steps. *See supra* Section VI.B.16[14.1]-[14.2a]; Ex. C, ¶ 235. The rest of this claim limitation is substantially similar to claim 6. *See* Section VI.B.11; Ex. C, ¶ 235. *Bauer* and *Wyman* render

obvious this limitation for the same reasons discussed in connection with claim 6. *Supra* Section VI.B.11; Ex. C, ¶ 235.

20. **Claim 18: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to implement an error routine that inhibits operation of the computing node when an incompatibility between the computing node and the computing cluster is detected.”**

As discussed for claim 14, *Bauer* and *Wyman* render obvious “[a] computer program product comprising logic instructions stored on a non-transitory computer-readable medium which, when executed by a computer processor, configure the processor” to perform various steps. *See supra* Section VI.B.16[14.1]-[14.2a]; Ex. C, ¶ 236. The rest of this limitation is substantially similar to claim 7. *See* Section VI.B.10; Ex. C, ¶ 236. *Bauer* and *Wyman* render obvious this limitation for the same reasons discussed in connection with claim 7. *Supra* Section VI.B.10; Ex. C, ¶ 236

C. Ground 3: *Baratti* either anticipates claims 2 and 15, or it renders them obvious in view of *Bauer* or *Wyman*.

Claims 2 and 15 require storing license information for a computing cluster in memory associated with a computing node by retrieving it from an external licensing server.

Baratti teaches that the licensing information stored in the servers is usually created by the servers “processing license certificate files, which are provided by the software vendor.” PA-1, [1:42-45]. Further, *Baratti* teaches that the “[l]icense certificate files may contain some encryption or checksum information that allow the license server to verify their authenticity and integrity.” *Id.*, [1:51-54].

A person of ordinary skill in the art at the time the ’416 patent application was filed, would understand *Baratti*’s disclosure that the licensing information is provided to the servers by the

software vendor to mean that servers retrieve this information from the vendor via an external server. Ex. C, ¶ 239.

But even if providing licensing information from a remote server were not inherent in *Baratti*, it would be obvious. By April 2006, this idea was ubiquitous. Ex. C, ¶ 240. For example, *Bauer* and *Wyman*, which both address licensing management in clustered computing networks, and which solve the same problems addressed by *Baratti* and the '416 patent, disclose that delivery of software products can be done faster, cheaper, and more conveniently over the Internet from an external server than by using the old ways of physically distributing licensing keys. Ex. C, ¶ 240; *see, e.g.*, PA-2, [0044]; PA-3, [10:30-57].

For a reference to be proper for use in an obviousness rejection under 35 U.S.C. 103, the reference must be analogous art to the claimed invention. *In re Bigio*, 381 F.3d 1320, 1325 (Fed. Cir. 2004). A reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). Note that "same field of endeavor" and "reasonably pertinent" are two separate tests for establishing analogous art; it is not necessary for a reference to fulfill both tests in order to qualify as analogous art. *See Bigio*, 381 F.3d at 1325.

As Dr. Weissman explains in his declaration, *Wyman* is analogous art because it is from the same field of endeavor of the '416 patent, *Baratti*, and *Bauer* and because they all solve essentially the same problem: how to efficiently license nodes in clustered computer networks. Ex. C, ¶ 242.

1. **Claim 2: “The method of claim 1, wherein storing license information for a computing cluster in a memory module associated with a computing node comprises retrieving licensing information from an external licensing server.”**
2. **Claim 15: “The computer program product of claim 14, further comprising logic instructions which, when executed by the processor, configure the processor to retrieve licensing information from an external licensing server.”**

Baratti in view of *Bauer* or *Wyman* renders obvious the limitations recited in claims 2 and 15; specifically, receiving license information from an external licensing server. Ex. C, ¶ 243. *Baratti* teaches that the licensing information stored in the servers is usually created by the servers “processing license certificate files, which are provided by the software vendor” and that the “[l]icense certificate files may contain some encryption or checksum information that allow the license server to verify their authenticity and integrity.” PA-1, [1:42-54].

While *Baratti* does not expressly state that the licensing information is retrieved from an external licensing server, this would have been inherent in *Baratti* or at least obvious to an ordinary artisan in view of *Bauer* or *Wyman*. Ex. C, ¶ 244. In April 2006, software was commonly sold, purchased, and managed through on-line websites. *Id.* A necessary part of authenticating networking software was to send licensing information over the Internet to a network administrator from a server operated by (on behalf of) the seller of the software. *Id.*

Bauer and *Wyman* disclose methods of issuing licensing information to a network administrator over the Internet. *Id.*, ¶ 245. For example, as discussed above in Section VI.B.12[9.4a], *Bauer* describes the cluster manager receiving a “license via the network . . . from a manufacturer of the computer system 100 or from a licensor of the associated resources.” PA-2, [0044]. And, as discussed above in Section VI.B.12[9.4a], *Wyman* also discloses receiving a license over a communications link from a “distributor who has received the authority to grant

licenses from the producer or owner.” PA-3, [10:30-57]. This requires the license to originate from an external server. Ex. C, ¶ 245.

It would have been obvious to an ordinary artisan to issue *Baratti*’s licensing certificates from an external server, as was commonly done in 2006 and as taught by *Bauer* and *Wyman*, because that would be a much faster, cheaper, and more convenient way to issue the license key in comparison to other known methods, such as by “physical delivery by disks or CD ROMs.” Ex. C, ¶ 246; *see also, e.g.*, PA-3, [10:49].

All three references are from the same field of endeavor, license management in clustered computer networks, and address the same problem, how to dynamically add and remove computers to and from a cluster in response to licensing requirements, network usage, and equipment failure. Ex. C, ¶ 247. Moreover, an ordinary artisan would have expected to succeed in delivering licensing information over the internet, as by April 2006, such operations were considered routine in the art of software licensing. *Id.*

D. Ground 4: *Baratti* in view of *Hughes* renders obvious claim 13.

1. Overview of *Hughes*

Hughes relates to licensing management and discloses a licensing system described as intended for incorporation into Microsoft Office. PA-4, ¶¶ [0014], [0024]. The system of *Hughes* creates a license file for a combination of hardware (a computer) and software (a software product running on the computer). *Id.*, ¶¶ [0015]-[0016], [0024]. The license file is checked whenever the software starts to determine whether the current combination of software and hardware sufficiently matches the combination specified by the license file. *Id.*, ¶ [0024], [0043], [0047], [0050]. When the current combination of software and hardware does not sufficiently match the specified

combination, the software cannot run on the current hardware. *Id.*, ¶ [0024], [0043], [0047], [0051].

Hughes discloses a process for generating the license file. PA-4, ¶¶ [0035]-[0040]. This process can be used to activate electronically delivered programs and stored in memory, and can be performed in distributed computing environments. *Id.*, ¶¶ [0027], [0032]. The user enters a CD key. *Id.*, ¶¶ [0033]. The computer automatically sends the CD key (or a product ID included in or derived from the CD key) and a hardware ID representing the current hardware over a network to an activation server. *Id.*, ¶¶ [0035]-[0040]. The computer then receives the license file in response. *Id.*, ¶ [0041].

Hughes explains that the disclosed licensing system beneficially prevents unauthorized copying of software onto another computer, while enabling customers to upgrade hardware components without requiring additional licenses. PA-4, ¶¶ [0046]-[0047].

2. Motivation to Combine *Baratti* and *Hughes*

In order for a reference to be proper for use in an obviousness rejection under 35 U.S.C. 103, the reference must be analogous art to the claimed invention. *In re Bigio*, 381 F.3d 1320, 1325 (Fed. Cir. 2004). A reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). Note that "same field of endeavor" and "reasonably pertinent" are two separate tests for establishing analogous art; it is not necessary for a reference to fulfill both tests in order to qualify as analogous art. *See Bigio*, 381 F.3d at 1325.

As Dr. Weissman explains in his declaration, *Hughes* is analogous art because it is both from the same field of endeavor of the '416 patent and *Baratti* and because it addresses, at least

with respect to the subject matter of claim 13, a similar problem: licensing of a particular software/hardware combination. Ex. C, ¶ 252.

The ordinary artisan would have found the licensing system of *Hughes* a predictable, easy-to-implement extension of the licensing described in *Baratti*. *Baratti* discloses licensing combinations of hardware (servers, PCs) and programs, by using encrypted licensing certificates obtained from a vendor. PA-1, [1:42-54]. *Baratti*'s existing licensing process could be straightforwardly extended through the automatic transmission of a validated license key (to decrypt the certificate) over a network to an activation server. Ex. C, ¶ 253. A license key could be received in response from the activation server and used to control activation of the software on *Baratti*'s servers. Ex. C, ¶ 253. This extension of the existing *Baratti* license process would beneficially prevent unauthorized copying of software onto another server, while enabling customers to add servers without requiring additional licenses. *Id.*

3. Claim 13: “The computing system of claim 10, wherein the management module provides a license key to and receives licensing information from an external licensing server.”

As shown herein, the only arguable difference between *Baratti* and claim 13 is the idea of providing a license key to the servers from an external licensing server. Ex. C, ¶ 254. This additional subject matter is clearly taught in *Hughes*. Ex. C, ¶ 254.

As described above, *Baratti* includes embodiments in which every node (server) has software stored in memory that receives an encrypted licensing certificate that the server processes to obtain licensing information, i.e., every node has the claimed “management module.” Section VI.A[9.4a]. *Hughes* discloses providing a product ID (constituting the license key) over a network to an activation server (constituting the external licensing server). PA-4, ¶¶ [0035]-[0040]. The computer then receives the license file (constituting the licensing information). PA-4, ¶ [0041].

It would have been obvious to the ordinary artisan to perform the licensing process of *Hughes* using the software of *Baratti* stored in memory that ensures the node receives licensing information in a secure manner. Ex. C, ¶ 256. This simple extension of the licensing process of *Baratti* would beneficially prevent unauthorized copying of software onto another, unlicensed, server, while enabling customers to upgrade the system without requiring additional licenses. Ex. C, ¶ 256.

Therefore, *Baratti* in view of *Hughes* teaches the cluster manager providing a message to the licensing server and receiving the license document from the licensing server. Ex. C, ¶ 257.

E. Ground 5: *Bauer* and *Wyman* in view of *Hughes* renders obvious claim 13.

1. Overview of *Hughes*

As described above, *Hughes* relates to licensing management and discloses a licensing system described as intended for incorporation into Microsoft Office. *Supra* Section VI.D.1.

2. Motivation to Combine *Bauer*, *Wyman*, and *Hughes*

In order for a reference to be proper for use in an obviousness rejection under 35 U.S.C. 103, the reference must be analogous art to the claimed invention. *In re Bigio*, 381 F.3d 1320, 1325 (Fed. Cir. 2004). A reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). Note that "same field of endeavor" and "reasonably pertinent" are two separate tests for establishing analogous art; it is not necessary for a reference to fulfill both tests in order to qualify as analogous art. *See Bigio*, 381 F.3d at 1325.

As Dr. Weissman explains in his declaration, *Hughes* is analogous art because it is both from the same field of endeavor of the '416 patent, *Bauer*, and *Wyman* and because it addresses, at least with respect to the subject matter of claim 13, a similar licensing problem. Ex. C, ¶ 260.

The ordinary artisan would have found the licensing system of *Hughes* a predictable, easy-to-implement extension of the licensing processes described in *Bauer* and *Wyman*. *Bauer* and *Wyman* both disclose methods of issuing a licensing information to a network administrator over the Internet. *Id.*, ¶ 261. For example, as discussed above in Section VI.B.12[9.4a], *Bauer* describes the cluster manager receiving a “license via the network . . . from a manufacturer of the computer system 100 or from a licensor of the associated resources.” PA-2, [0044]. And, as discussed above in Section VI.B.12[9.4a], *Wyman* also discloses receiving a license over a communications link from a “distributor who has received the authority to grant licenses from the producer or owner.” PA-3, 10:30-57. This requires the license to originate from an external server. Ex. C, ¶ 261.

The licensing processes described in *Bauer* and *Wyman* could be extended by receiving the license file of *Hughes* in response to the automatic transmission of a license key (and hardware information and for a node) over a network to an activation server. Ex. C, ¶ 262. A licensing file could be received in response from the activation server and used to control activation of the software on the node. Ex. C, ¶ 262. This extension of the existing *Bauer* and *Wyman* license processes would beneficially prevent unauthorized copying of software onto another cluster, while enabling customers to upgrade cluster components without requiring additional licenses. Ex. C, ¶ 262.

3. Claim 13: “The computing system of claim 10, wherein the management module provides a license key to and receives licensing information from an external licensing server.”

As shown herein, the only arguable difference between *Bauer* and *Wyman* and claims 13 is the idea of providing a license key to and receiving licensing information from an external licensing server. Ex. C, ¶ 263. This additional subject matter is clearly taught in *Hughes*. Ex. C, ¶ 263

As described above, *Bauer* teaches the cluster manager that receives a license to a number of resources in the cluster and saves the number of licensed resources in the number of licenses 340 in the cluster resource status 144. PA-2, [0044], [0045]. *Hughes* discloses providing a product ID (constituting the license key) to over a network to an activation server (constituting the external licensing server). PA-4, ¶¶[0035]-[0040]. The computer then receives the license file (constituting the licensing information). PA-4, ¶ [0041].

It would have been obvious to the ordinary artisan to perform the licensing process of *Hughes* using the cluster manager of *Bauer*, already disclosed as receiving license information. Ex. C, ¶ 265. This simple extension of the licensing process of *Bauer* would beneficially prevent unauthorized copying of software onto another cluster, while enabling customers to upgrade cluster components without requiring additional licenses. Ex. C, ¶ 265.

Therefore, *Bauer* and *Wyman* in view of *Hughes* teaches the cluster manager providing a message to the licensing server and receiving the license document from the licensing server. Ex. C, ¶ 266.

VII. CONCLUSION

For the foregoing reasons, NetApp requests reexamination and cancellation of claims 1-18 of the '416 patent.

NetApp is willing to provide appropriate assistance to permit the Examiner to address and decide the issues presented by this Request. As the M.P.E.P. explains, the Examiner may, when appropriate, cut and paste claim charts or other material within the Request to incorporate them within the body of an Office Action. M.P.E.P. § 2262. NetApp is, thus, through the undersigned counsel, available to provide the Examiner with a digital copy of this Request, or any portion of it, in response to a request by email or phone. NetApp also understands that the Examiner may, in appropriate circumstances, set forth specific rejections in an Office Action and incorporate by reference NetApp's reasons for the proposed rejections, if the Examiner agrees with the proposed rejections and reasons supporting them.

If there is any additional fee due in connection with this filing that is not already submitted herewith, please charge the fee to Deposit Account No. 06-0916.

Respectfully submitted,
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP

Dated: November 14, 2025

By: /Cory C. Bell/
Cory C. Bell
Reg. No. 75,096

**Attachments: (1) Certificate of Service to Patent Owner
(2) Information Disclosure Statement Form PTO/SB/08
(3) Exhibits listed on page iii of this Request**