

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of: Hoover et al.
U.S. Patent No.: 8,370,416 Attorney Docket No.: 39898-0049IP1
Issue Date: February 5, 2013
Appl. Serial No.: 11/411,732
Filing Date: April 26, 2006
Title: COMPATIBILITY ENFORCEMENT IN CLUSTERED
COMPUTING SYSTEMS

DECLARATION OF DR. CHANDRAJIT BAJAJ

TABLE OF CONTENTS

I.	INTRODUCTION	9
II.	QUALIFICATIONS	9
III.	LEGAL PRINCIPLES	14
	A. Person of Ordinary Skill in the Art	15
	B. Obviousness.....	16
IV.	LEVEL OF ORDINARY SKILL IN THE ART	19
V.	INTERPRETATIONS OF THE '416 PATENT CLAIMS AT ISSUE	20
VI.	MATERIALS CONSIDERED	21
VII.	OVERVIEW OF CONCLUSIONS FORMED.....	23
VIII.	THE '416 PATENT	23
	A. Overview	23
	B. File History.....	27
IX.	OVERVIEW AND COMBINATIONS OF PRIOR ART REFERENCES	31
	A. Sparks (FEDEX-1005)	31
	B. Bram (FEDEX-1006)	38
	C. The Sparks-Bram Combination.....	41
X.	GROUND 1: SPARKS AND BRAM	50
	A. Claim 1	50
	B. Claim 2	84

C.	Claim 3	89
D.	Claim 4	90
E.	Claim 5	94
F.	Claim 6	100
G.	Claim 7	106
H.	Claim 8	108
I.	Claim 9	110
J.	Claim 10	119
K.	Claim 11	122
L.	Claim 12	124
M.	Claim 13	125
N.	Claim 14	129
O.	Claim 15	132
P.	Claim 16	132
Q.	Claim 17	132
R.	Claim 18	133
XI.	CONCLUSION	134

LISTING OF CLAIMS

Claim 1	
[1pre]	A method of creating a clustered computing system, comprising:
[1a-i]	storing license information for a computing cluster in a memory module associated with a computing node,
[1a-ii]	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;
[1b]	initializing the computing cluster in a first computing node;
[1c]	adding one or more available computing nodes to the computing cluster; and
[1d]	activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.
Claim 2	
[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.
Claim 3	
[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.

Claim 4	
[4]	The method of claim 1, wherein: the node license parameters identify one or more bundle-types compatible with the computing node.
Claim 5	
[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.
Claim 6	
[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.
Claim 7	
[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.
Claim 8	
[8]	The method of claim 1, further comprising adding a computing node to the activated computing cluster.
Claim 9	
[9pre]	A computing system comprising at least two computing nodes, wherein at least one of the computing nodes comprises:
[9a-i]	a cluster module stored on a non-transitory computer readable medium, the cluster module configured to manage compatibility operations associated with a computing cluster,

[9a-ii]	the cluster module initializing a computing cluster in the first computing node,
[9a-iii]	the cluster module adding one or more available computing nodes to the computing cluster;
[9b-i]	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,
[9b-ii]	the node module storing license information for the computing cluster,
[9b-iii]	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
[9c-i]	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,
[9c-ii]	the management module activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.
Claim 10	
[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.
Claim 11	
[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.

Claim 12	
[12]	The computing system of claim 10, further comprising an add node module to add a computing node to the clustered computing system.
Claim 13	
[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.
Claim 14	
[14pre]	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:
[14a-i]	store license information for a computing cluster in a memory module associated with a computing node,
[14a-ii]	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;
[14b]	initialize the computing cluster in a first computing node;
[14c]	add one or more available computing nodes to the computing cluster; and
[14d]	activate the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.
Claim 15	
[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.

Claim 16	
[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.
Claim 17	
[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.
Claim 18	
[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.

I, Chandrajit Bajaj, PhD, declare as follows:

I. INTRODUCTION

1. I have been retained as a technical expert by counsel on behalf of FedEx Corporation (“FedEx”/“Petitioner”). I understand that FedEx is requesting that the Patent Trial and Appeal Board institute an *inter partes* review (“IPR”) proceeding with respect to claims 1–18 (“Challenged Claims”) of U.S. Patent 8,370,416 (“the ’416 Patent”) (FEDEX-1001).

2. I have been asked to provide my independent analysis of the ’416 Patent in light of certain prior art publications, and I have done so based on my personal knowledge and experience.

3. I am not, and never have been, an employee of FedEx. I received no compensation for this Declaration beyond my normal hourly compensation based on my time actually spent analyzing the ’416 Patent, the prior art publications cited below, and various issues related thereto. I will not receive any added compensation based on the outcome of any IPR or other proceeding involving the ’416 Patent.

II. QUALIFICATIONS

4. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field. My education and experience are described more fully in my curriculum vitae (“CV”) which is attached as Exhibit FEDEX-1004

of the IPR petition. For ease of reference, I have highlighted certain information below.

5. I received my Ph.D. and M.Sc. degrees in computer science from Cornell University in 1984 and 1983, respectively. I also received my Bachelor of Technology degree in electrical engineering from the Indian Institute of Technology Delhi in 1980. During my college studies from 1975 to 1980, I focused on designing and using programmable micro-controllers based on commodity microprocessors, as well as very large scale integrated (VLSI) circuit design. I took specialized courses including graduate courses in computer algorithms and mathematical programming. As part of my doctoral research at Cornell from 1980 to 1984, I worked on developing and implementing methods for large combinatorial and geometric optimization problems.

6. From 1984 to 1997, I was a professor of computer sciences at Purdue University while also serving as the director of the Image Analysis and Visualization Center at the university. From 1990 to 1991, I was a visiting associate professor of computer sciences at Cornell University. As part of my work at Purdue and Cornell, I researched, taught, and developed high-performance, distributed, and parallel computer graphics and geometric modeling systems. I also developed new algorithms to model, simulate, and visualize natural and synthetic 3D and 4D virtual environments, combining computational scanning, and rapid scene generation

algorithms deployed on distributed, collaborative, and parallel architectures. Exemplary publications during this time include: “Shastra: Multimedia Collaborative Design Environment,” “SHASTRA - An Architecture for Development of Collaborative Applications,” “Collaborative Multimedia in SHASTRA,” “Distributed and Collaborative Synthetic Environments,” etc.

7. Since 1997, I have been a professor of computer sciences and the Computational Applied Mathematics Chair in Visualization, at the University of Texas at Austin. During this time, I have researched, taught, and used high-performance, distributed, and parallel computer graphics systems including commodity off-the-shelf (COTS) clusters systems. I have also been an active member of a team developing hardware and software technology that allowed multiple computers with multiple programmable graphics cards (GPUs) to simultaneously and synchronously display to large multi-screen immersive displays. Our team called this multi-displays and many-graphics networked processors, cluster programming technology the UT Meta-Buffer solution. One of the peer-reviewed publications that resulted from this is titled “Active Visualization in a Multi-display Immersive Environment.”

8. Much of my past and current work involves issues relating to 2D and 3D image capture, and reconstruction, compression, image and geometric data processing, modeling, and quantitative analysis. I have participated in the design and

use of several computer systems spanning handhelds, laptops, and graphics workstations to PC/Linux clusters, as well as very large memory supercomputers for capturing, modeling, and displaying virtual and scientific phenomena. My experience with computational modeling and displaying computer graphics imagery touches many fields, such as interactive games, molecular, biomedical and industrial diagnostics, oil and gas exploration, geology, cosmology, and military industries.

9. During this time, I authored or co-authored about 300 papers, book chapters, conference papers, and technical reports, a select sample of which include the following papers directed at high-performance, distributed, and parallel computer systems, or clustered/GPU systems (see my Curriculum Vitae): (i) “Parallel Accelerated Iso-contouring for Out-of-Core Visualization” – one of the first demonstrations of moving heavy compute to a bank of off-board commodity GPUs to free the main CPU and graphics card for display work; (ii) “Compression-Based 3-D Texture Mapping for Real-Time Rendering” - addressing GPU texture-memory bandwidth limits and efficient texture management; (iii) “Scalable Isosurface Visualization of Massive Datasets on COTS Clusters” - detailing strategies to minimize PCI-e bottlenecks while streaming volumetric data to multiple GPUs; (iv) “SIMD Optimization of Linear Expressions for Programmable Graphics Hardware” - work on fine-grained instruction scheduling inside shader ALUs and (v). “Accelerated Molecular Mechanical and Solvation Energetics on Multicore

CPU's and Many-Core GPU's" - work on using cluster tiers of CPU's and GPU's for highest performance calculations needed for protein folding. *See also* "Active Visualization in a Multidisplay Immersive Environment" and "A Cluster Based Emulator for Multidisplay, Multiresolution Parallel Image Compositing".

10. I am knowledgeable about and have experience in developing both parallel hardware and software in the field of high-performance, distributed, and parallel computer systems or clusters, including for many core GPUs and multi-core CPU cluster systems. Specifically, I led a team that built the first data intensive and display intensive visualization commodity cluster with 128 workstations interconnected with InfiniBand and used in driving immersive displays and visualization walls.

11. As outlined in my *Curriculum Vitae*, I have authored approximately 35 peer-reviewed book chapters, and 173 peer-reviewed journal and 162 peer-reviewed conference publications. I am also a co-inventor of U.S. Patent 6,438,266, titled "Encoding Images of 3-D Objects with Improved Rendering Time and Transmission Processes," issued on August 20, 2002.

12. I have written and edited four books, on topics ranging from image processing, geometric modeling, and visualization techniques to algebraic geometry and its applications. I have given 234 invited speaker, plenary and keynote talks on these topics. I am a Fellow of the American Association for the Advancement of

Science (AAAS), a Fellow of the Institute of Electrical and Electronics Engineers (IEEE), a Fellow of the Society of Industrial and Applied Mathematics (SIAM), and a Fellow of the Association of Computing Machinery (ACM), which is the world's largest education and scientific computing society. ACM Fellow is ACM's most prestigious member grade and recognizes the top 1% of ACM members for their outstanding accomplishments in computing and information technology and/or outstanding service to ACM and the larger computing community.

13. A detailed list of my other professional activities, memberships, and speaking engagements is included in my CV.

14. Based on my above-described three decades of experience in high-performance computing, distributed/clustered computer systems, geometric modeling, computer graphics and visualization technologies, and the acceptance of my publications and professional recognition by societies in my field, I believe that I am qualified to be an expert in distributed/clustered computing systems (including compatibility and licensing enforcement in distributed/clustered environments).

III. LEGAL PRINCIPLES

15. In forming my opinions and considering the subject matter of the '416 Patent and its claims in light of the prior art, I am relying on certain legal principles

that counsel in this case explained to me. My understanding of these concepts is summarized below.

16. I understand that the claims define the invention. I also understand that an unpatentability analysis is a two-step process. First, the claims of the patent are construed to determine their meaning and scope. Second, after the claims are construed, the content of the prior art is compared to the construed claims.

17. I understand that a claimed invention is only patentable when it is new, useful, and non-obvious in light of the “prior art.” That is, the invention, as defined by the claims of the patent, must not be anticipated or rendered obvious by the prior art.

A. Person of Ordinary Skill in the Art

18. I understand that the factors considered in determining the ordinary level of skill in a field of art include: the level of education and experience of persons working in the field; the types of problems encountered in the field; the teachings of the prior art regarding solutions to such problems; and the sophistication of the technology at the time of the alleged invention. I understand that a person of ordinary skill in the art (“POSITA”) is not a specific real individual, but rather is a hypothetical individual having the qualities reflected by the factors above and knowledge of all relevant prior art references, including the references I cite below.

B. Obviousness

19. I understand that a patent claim is invalid as obvious if the claimed invention would have been obvious to a POSITA at the time the claimed invention was made. This means that even if all of the elements of the claim cannot be found in a single prior art reference that would anticipate the claim, a person of ordinary skill in the field who knew about all the prior art would have come up with the claimed invention. I understand that in an obviousness determination, the person of ordinary skill in the art is presumed to have knowledge of all material prior art. I understand that whether a claim is obvious is based upon the determination of several factual issues.

20. I understand that obviousness is a determination of law based on underlying determinations of fact. I understand that these factual determinations include the scope and content of the prior art, the level of ordinary skill in the art, the differences between the claimed invention and the prior art, and secondary considerations of non-obviousness.

21. In considering obviousness, I understand that one must determine the scope and content of the prior art. I understand that, in order to be considered as prior art to a patent being considered, a prior art reference must be reasonably related to the claimed invention of that patent. A reference is reasonably related if it is in the

same field as the claimed invention or is from another field to which a person of ordinary skill in the art would look to solve a known problem.

22. I understand that one must determine what differences, if any, existed between the claimed invention and the prior art.

23. I understand that a patent claim composed of several elements is not proved obvious merely by demonstrating that each of its elements was independently known in the prior art. In evaluating whether such a claim would have been obvious, one may consider whether a reason has been identified that would have prompted a person of ordinary skill in the art to combine the elements or concepts from the prior art in the same way as in the claimed invention. There is no single way to define the line between true inventiveness on the one hand (which is patentable) and the application of common sense and ordinary skill to solve a problem on the other hand (which is not patentable). For example, market forces or other design incentives may be what precipitated a change, rather than true inventiveness.

24. I understand that whether a prior art reference renders a patent claim unpatentable as obvious is determined from the perspective of a person of ordinary skill in the art at the time of the alleged invention. I have been told that there is no requirement that the prior art contain an express suggestion to combine known elements to achieve the claimed invention, but a suggestion to combine known elements to achieve the claimed invention may come from the prior art, as filtered

through the knowledge of one skilled in the art. In addition, I have been told that the inferences and creative steps a person of ordinary skill in the art would employ are also relevant to the determination of obviousness.

25. I understand that there is no rigid rule that a reference or combination of references must contain a “teaching, suggestion, or motivation” to combine references. But I also understand that the “teaching, suggestion, or motivation” test can be a useful guide in establishing a rationale for combining elements of the prior art. I have been told that this test poses the question as to whether there is an express or implied teaching, suggestion, or motivation to combine prior art elements in a way that realizes the claimed invention, and that it seeks to counter impermissible hindsight analysis.

26. I understand that one may consider, e.g., whether (1) the change was merely the predictable result of using prior art elements according to their known functions, or whether it was the result of true inventiveness; (2) there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; (3) the claimed innovation applies a known technique that had been used to improve a similar device or method in a similar way; (4) the claimed invention would have been obvious to try, meaning that the claimed innovation was one of a relatively small number of possible approaches to the problem with a reasonable expectation of success by those skilled in the art; (5) the

invention merely substituted one known element for another known element in order to obtain predictable results; (6) the invention merely applies a known technique to a known device, method, or product to yield predictable results; or (7) known work in the field may have prompted variations of use of the same inventions in the same or different fields due to market forces or design incentives that would have been predictable to a person of ordinary skill in the art.

27. I understand that any assertion of secondary considerations of non-obviousness must be accompanied by a nexus between the merits of the invention and the evidence offered.

IV. LEVEL OF ORDINARY SKILL IN THE ART

28. In rendering the opinions set forth in this declaration, I was asked to consider the patent claims and the prior art through the eyes of one of ordinary skill at the time of the alleged invention, which I understand is asserted to be April 26, 2006 (“Critical Date”)—the filing date of the application which issued as the ’416 Patent.

29. Based on my knowledge and experience in the field and my review of the ’416 Patent and its file history, I believe that a POSITA would have had (1) at least a master’s of science degree in computer science, computer engineering, or a related field, and (2) at least two years of professional experience involving the study

and/or development of distributed, clustered, and/or grid computer systems. Additional education could substitute for experience and vice versa.

30. My analysis and conclusions expressed in this Declaration are based on the perspective of a POSITA having this level of knowledge and skill. My education, technical expertise and personal knowledge discussed in Section II shows that I meet the qualifications of a POSITA.

V. INTERPRETATIONS OF THE '416 PATENT CLAIMS AT ISSUE

31. For purposes of my analysis in this IPR proceeding, I understand that the terms that appear in the claims of the '416 Patent should be interpreted according to their "ordinary and customary meaning," which is sometimes called the *Phillips* standard. In determining the ordinary and customary meaning, I understand that the words of a claim are first given their plain meaning that those words would have had to a POSITA at the time. I also understand that the structure of the claims, the specification, and the file history also may be used to better construe a claim insofar as the plain meaning of the claims cannot be understood. Moreover, I understand that even treatises and dictionaries may be used, albeit under limited circumstances, to determine the meaning attributed by a POSITA to a claim term at the time of filing. I have followed this approach in interpreting terms of the Challenged Claims herein.

VI. MATERIALS CONSIDERED

32. My analysis in this Declaration is based on my knowledge and experience. Based on my above-described qualifications in Section II, the Patent Trial and Appeal Board should consider me to be an expert in the field. Also, based on my experiences, I understand and know of the capabilities of persons of ordinary skill in this field.

33. In writing this Declaration, I have considered the following: my own knowledge and experience, including my own knowledge and experience gained from my work in the field; my experience in teaching those subjects; and my experience in working with others involved in those fields. In addition, I have analyzed the following publications and materials, in addition to other materials I cite in my declaration:

- U.S. Patent 8,370,416 (FEDEX-1001)
- File History of U.S. Patent 8,370,416 (FEDEX-1002)
- U.S. Patent 7,590,653 (“Sparks”) (FEDEX-1005)
- U.S. Publication 2006/0179058 (“Bram”) (FEDEX-1006)
- Resources for Business, Business Software Alliance, 2004 retrieved from <https://web.archive.org/web/20040612091712/http://www.bsa.org/usa/antipiracy/Resources-for-Business.cfm> (FEDEX-1007)

- Piracy and The Law, Business Software Alliance, 2004 retrieved from <https://web.archive.org/web/20040614221833/http://www.bsa.org/usa/antipiracy/Piracy-and-the-Law.cfm> (FEDEX-1008)
- Microsoft Computer Dictionary, Fifth Edition, 2002 (FEDEX-1009)
- The New Penguin Dictionary of Computing, 2001 (FEDEX-1010)
- FLEXlm End Users Guide, Version 9.5, August 2004 (FEDEX-1011)
- Administering Platform LSF, Version 5.1, February 2003 (FEDEX-1012)

34. Note that my citations to non-patent literature throughout this Declaration reference the absolute page number added to the exhibit (as opposed to the original pagination of the document). For patent literature, I've used the column/line numbers or paragraph numbers. Note also that, unless I've indicated otherwise, all emphasis in any quoted text are ones I have added.

35. Although I cite and quote to selected portions of various references in this Declaration, the reader should understand that these citations are representative examples. A POSITA would have viewed each reference in its entirety and in combination with other references. Accordingly, I intend the references identified in this Declaration to be viewed as incorporated in their entireties.

VII. OVERVIEW OF CONCLUSIONS FORMED

36. This expert declaration explains the conclusions that I have formed based on my analysis. To summarize those conclusions, based upon my knowledge and experience and my review of the prior art publications listed above, it is my opinion that Claims 1-18 of the '416 Patent are obvious over Sparks in view of Bram.

VIII. THE '416 PATENT

A. Overview

37. The '416 Patent is entitled "COMPATIBILITY ENFORCEMENT IN CLUSTERED COMPUTING SYSTEMS." FEDEX-1001, Cover. Under the "TECHNICAL FIELD" heading, the '416 Patent states that it "relates to electronic computing, and more particularly to compatibility enforcement in clustered computing systems." *Id.*, 1:6-8. The abstract of the '416 Patent discloses:

In one embodiment a method of creating a clustered computing system, comprises 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 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.

FEDEX-1001, Abstract.

38. The background section of the '416 Patent describes a clustered computing system and identifies issues purported with such systems.

The term “clustered computing system” refers to a computing system formed from a group of loosely coupled computing devices that cooperate to provide the appearance that the devices function as a single system. Clustering technology permits users to construct a variety of computing system configurations using elemental building blocks, and to change configurations or components of a clustered system as part of maintenance or update routines. By way of example, a high-capacity storage system may be constructed from a cluster of individual storage components. Similarly, a grid computer may be constructed from a cluster of individual computing components.

Computing systems and products created using clustering technology present particular technical issues. *Technical issues include ensuring compatibility between hardware, software and firmware operating on clustered devices and ensuring compliance with licensing terms associated with individual computing devices and with cluster configurations.*

FEDEX-1001, 1:11-30.

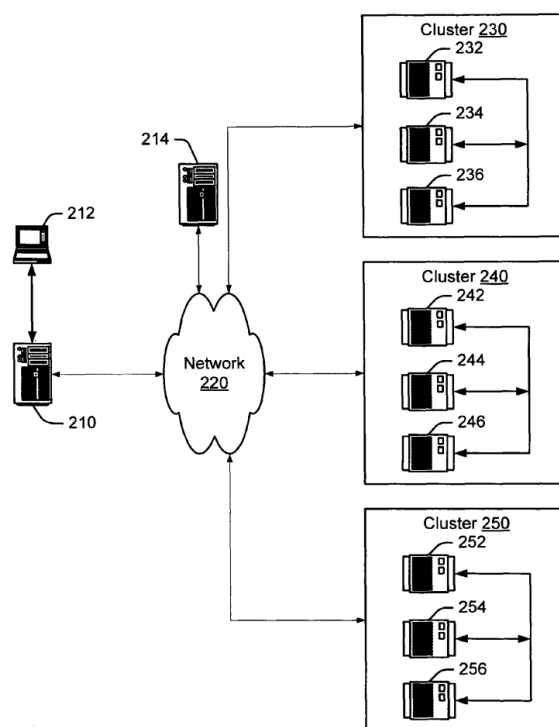
39. The '416 Patent describes a clustered computing system “may be adapted to implement compatibility enforcement between components in the system” including “compatibility between hardware, software, firmware, and/or licensing requirements.” *Id.*, 5:40-6:3.

In embodiments, a clustered computing system such as the clustered storage devices depicted in FIGS. 1-3 may be adapted to implement compatibility enforcement between components in the system. As used herein, compatibility enforcement may include compatibility between hardware, software, firmware, and/or licensing requirements. In one embodiment, a clustered computing system implements a licensing scheme that ensures licensing compliance at both the component level

and at the system level. *The licensing scheme may implement a multi-tiered (or bundled) license program.* The first part of the license scheme defines *a permissible size range for a cluster.* In one embodiment, *clusters may be categorized into multiple size ranges, referred to herein as bundle-types.* In alternate embodiments, 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. *A bundle may be sold as a set or group of node or grid types to meet a specific market or function.* For example, a product category may include one or more bundle sizes that served a single purpose such as, e.g., array/LUN purpose. *Bundles may have a specific homogeneous purpose along with standard grid parameters such as, e.g., right to use, maximum and minimum node counts, and a specific array type functional brand.* Other bundles may have different characteristics within those listed above, for example, the purpose or function. A bundle may have multipurpose or multiple different single purposed nodes in one bundle. The second part of the license scheme permits the cluster to permanently associate a license and bundle-type with a defined number of nodes as they join the cluster.

FEDEX-1001, 5:40-6:3.

40. A clustered computing system can include one or more clusters, where each cluster includes one or more nodes. *Id.*, 4:39-5:8.



FEDEX-1001, Fig. 2.

41. Figure 5 depicts “operations in a method for creating an initial clustered computing system.” *Id.*, 6:55-8:23.

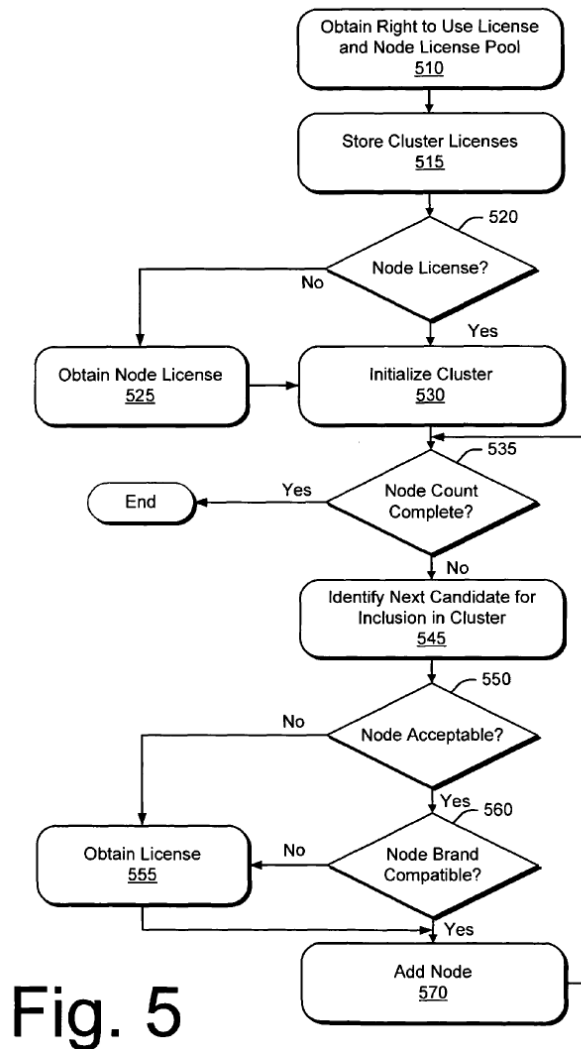


Fig. 5

FEDEX-1001, Fig. 5.

B. File History

42. As part of my preparation of this declaration, I reviewed the file history of the '416 Patent (FEDEX-1002). I understand that the '416 Patent issued on February 5, 2013 from U.S. Patent Application No. 11/411,732 ("the '732

application”). FEDEX-1001, Cover. The ’732 application was filed on April 26, 2006. *Id.*

43. The claims were initially rejected over Bauer (US 2006/0036894) or over a combination of Bauer and Yellai (US 2006/0106728). FEDEX-1002, pp. 620-630. Applicant made a few minor amendments and provided arguments to distinguish the claims over Bauer. *Id.*, pp. 602-615. The examiner maintained the prior art rejections. *Id.*, pp. 586-599. Applicant filed a response with a request for continued examination. *Id.*, pp. 458-471.

44. The examiner issued a new non-final rejection which continued the rejections over Bauer or in combination with Yellai. *Id.*, pp. 357-369. Applicant amended the claims by adding, to each of the independent claims, that the bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster. *Id.*, pp. 337-352.

1. (Currently Amended) 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 ~~characteristics~~ 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.

Id., p. 338. Applicant provided arguments to distinguish the claims over Bauer, again. *Id.*

45. The Examiner issued a final rejection over Bauer and Davidson (US 2005/0234846). *Id.*, pp. 321-333. The Examiner stated that “Davidson teaches a bundle-type parameter that identifies a size of the computer cluster.” *Id.* In combination with Bauer and Davidson, the examiner continued to used Yellai for a subset of the dependent claims. *Id.*

46. Applicant submitted an appeal brief. *Id.*, pp. 287-312. Applicant provided the following summary for claim 1:

With regard to independent claim 1, discussions of the recited features can be found at least in the below-cited locations of the specification and drawings. By way of example, claim 1 recites a method (*e.g.*, Fig. 5) of creating a clustered computing system, which method includes storing license information (*e.g.*, 515, Fig. 5) for a computing cluster in a memory module associated with a computing node. Specification, p. 15, l. 19 – p. 6, l. 3. 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 identify a characteristic of a computing node. *Id.*, p. 12, l. 12 – p. 13, l. 8. The computing cluster is initialized (*e.g.*, 530, Fig. 5) in a first computing node, and one or more available computing nodes are added (*e.g.*, 570, Fig. 5) to the computing cluster. *Id.*, p. 16, l. 15 – p. 18, l. 12. The computing cluster is activated when it includes a number of nodes that comply with the bundle-type parameter. *Id.*, p. 16, l. 15 – p. 17, l. 6; p. 2, ll. 10-12.

FEDEX-1002, p. 289. The applicant also provided arguments to distinguish over Bauer and Davidson. *Id.*, pp. 287-312.

47. In the reasons for allowance, the Examiner provided the following:

6. This communication warrants no examiner's reason for allowance, as applicant's reply makes evident the reason for allowance, satisfying the record as whole as required by rule 37 CFR 1.104 (e). In this case, the substance of applicant's appeal brief filed on 06/07/12 with respect to arguments that point out and make clear the reason claims are patentable over the prior art of record. Thus, the reason for allowance is in all probability evident from the record and no statement for examiner's reason for allowance is necessary (see MPEP 13202.14).

Id., pp. 262-274 (Notice of Allowance).

48. In addition to the references cited by the Examiner during prosecution, each of the claimed limitations was well-known in the art as further demonstrated by references like Sparks and Bram for the reasons I discuss herein. The file history

contains no indication that the Examiner had ever considered the prior art combinations advanced in the grounds of this Petition before the '732 application was allowed.

IX. OVERVIEW AND COMBINATIONS OF PRIOR ART REFERENCES

A. Sparks (FEDEX-1005)

49. Sparks is entitled “AUTOMATED DISCOVERY AND INVENTORY OF NODES WITHIN AN AUTONOMIC DISTRIBUTED COMPUTING SYSTEM.” FEDEX-1005, Cover. Sparks issued on September 15, 2009 from a patent application filed on March 2, 2005. *Id.* Under the “TECHNICAL FIELD” heading, Sparks states that “[t]he invention relates to computing environments and, more specifically, to distributed computing systems.” *Id.*, 1:7-8. The abstract of Sparks discloses:

A distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model. The model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of distributed computing system. The control node detects the addition of a node added to the network and automatically identifies attributes for the detected node.

FEDEX-1005, Abstract.

50. Both Sparks and the '416 Patent disclose a clustered computing system and techniques for its management. Sparks discloses for example:

In general, the invention is directed to *a distributed computing system that conforms to a multi-level, hierarchical organizational model*. One or more control nodes provide for the *efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model*.

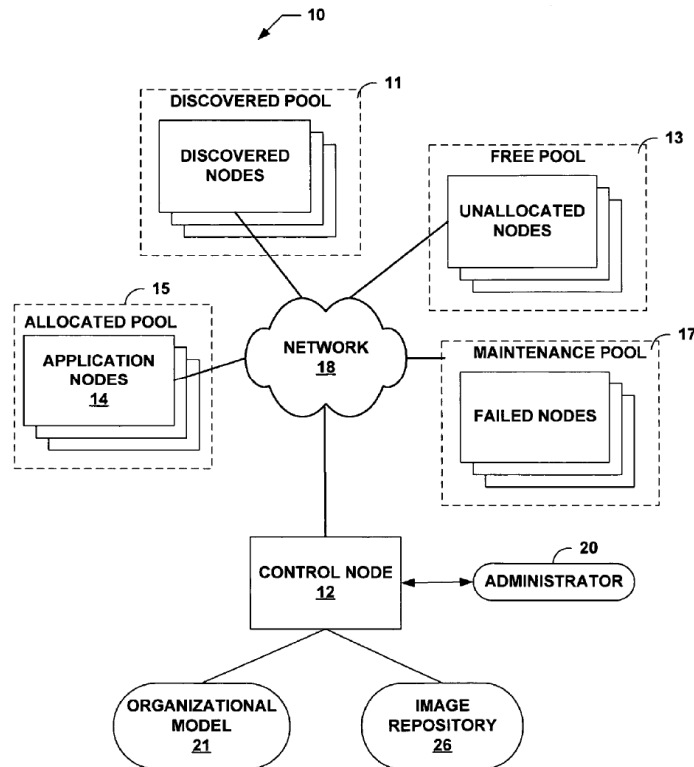
As described herein, the model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of the distributed computing system. *The control nodes are responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment*.

FEDEX-1005, 1:33-50.

51. Sparks describes a distributed computing system which includes computing nodes organized into one or more tiers.

FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. *Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide high-performance processing*.

FEDEX-1005, 3:7-11.



FEDEX-1005, Fig. 1.

In general, *distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes*. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

As used herein, the “fabric” level generally refers to the logical constructs that allow for definition, deployment, partitioning and management of distinct enterprise applications. In other words, fabric refers to the integrated set of hardware, system software and application software that can be “knitted” together to form a complete enterprise system. In general, the fabric level consists of two elements: fabric components or fabric payload. Control node 12 provides fabric management and fabric services as described herein.

In contrast, a “domain” is a logical abstraction for containment and

management within the fabric. The domain provides a logical unit of fabric allocation that enables the fabric to be partitioned amongst multiple uses, e.g. different business services.

Domains are comprised of *tiers, such as a 4-tier application model (web server, application server, business logic, persistence layer) or a single tier monolithic application*. Fabric domains contain the free pool of devices available for assignment to tiers.

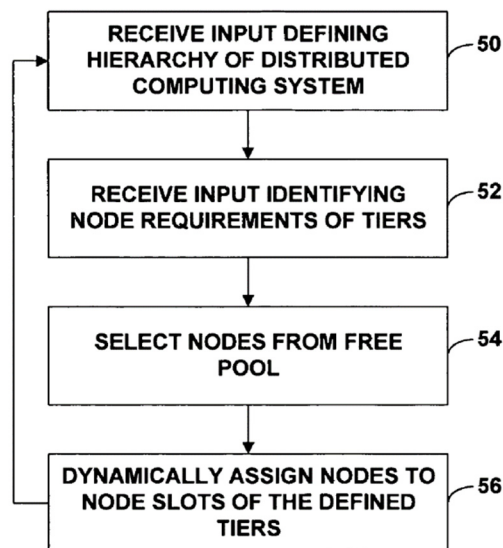
A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

The lowest level, the node level, includes the physical components of the fabric. This includes computing nodes that, as described above, provide operating environments for system applications and enterprise software applications. In addition, the node level may include network devices (e.g., Ethernet switches, load balancers and firewalls) used in creating the infrastructure of network 18. The node level may further include network storage nodes that are network connected to the fabric.

FEDEX-1005, 5:23-67.

52. As I discuss in more detail in the sections that follow, a POSITA would have recognized that Sparks discloses a clustered computing system, because a distributed computing system is an example of a clustered computing system. Also, each tier (including its computing nodes) in Sparks is likewise itself a computing cluster within Sparks's distributed computing system.

53. Sparks describes that a system administrator “may define policies regarding tier characteristics” as well as “specific node requirements for each tier of the fabric.” FEDEX-1005, 6:42-67. These characteristics/requirements are used to assign nodes to each tier. An example of this is shown in Figure 3 which is “a flow diagram that provides a high-level overview of the operation of control node 12 when configuring distributed computing system 10.” FEDEX-1005, 8:37-39.



FEDEX-1005, Fig. 3.

54. In the flow diagram of Figure 3, the “control node 12 receives input from a system administrator defining the hierarchical organization of distributed computing system 10 (50)” and “receive[s] input specifying node requirements of each of the tiers of the hierarchical model (52).” FEDEX-1005, 8:37-60. The control

node “assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes.”

Control node 12 dynamically *assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes* (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may *populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers*. Thus, *the computing nodes may be assigned using a “best fit” algorithm*.

FEDEX-1005, 9:3-13.

55. Figure 4 of Sparks provides additional details of node assignment and tier enablement.

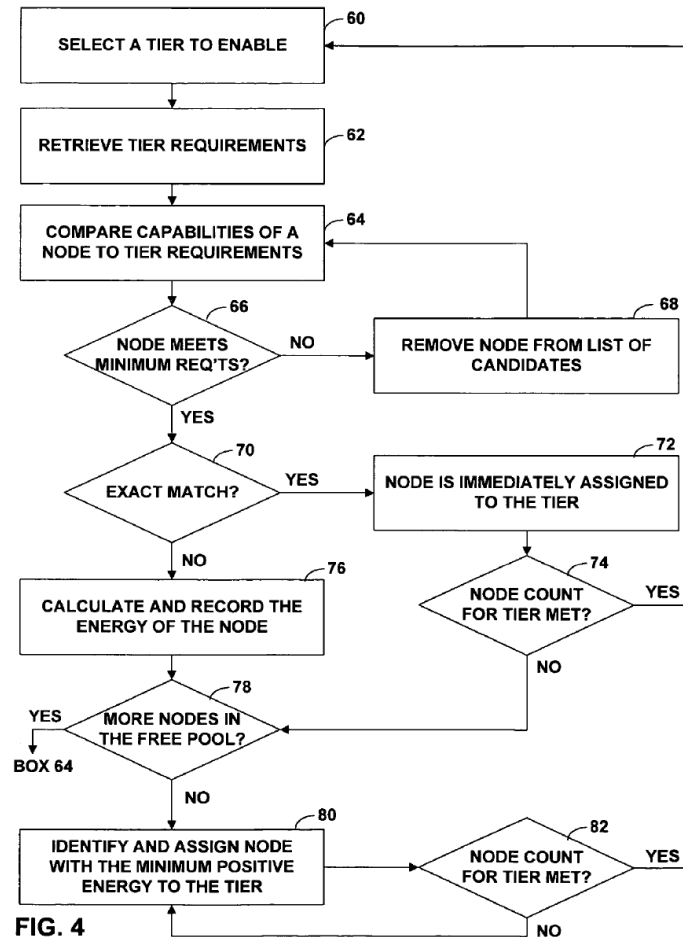
FIG. 4 is a flow diagram illustrating exemplary operation of control node 12 when *assigning computing nodes to node slots of tiers*. Initially, *control node 12 selects a tier to enable (60)*. As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, *initially select the tier with the highest priority and successively enable the tiers based on priority*.

FEDEX-1005, 9:14-21.

Control node 12 then determines whether the node count for the tier is met (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20. *When the node count for the tier is met, control node 12 selects the next tier to enable, e.g., the tier with*

the next largest priority, and repeats the process until all defined tiers are enabled, i.e., populated with application nodes (60).

FEDEX-1005, 9:52-62.



FEDEX-1005, Fig. 4. Sparks additionally discloses that “a user may request that a newly created tier come online” via enabling the tier.

Goal manager 312 may automatically formulate dependencies between goals or may allow a user to specify the dependencies. For example, a user may request that a newly created tier come online, and goal manager 312 may automatically generate a goal of harvesting a target number of nodes to enable the tier.

FEDEX-1005, 18:45–50.

B. Bram (FEDEX-1006)

56. Bram is entitled “METHODS AND SYSTEMS FOR LICENSING COMPUTER SOFTWARE.” FEDEX-1006, Cover. Bram published on August 10, 2006 from an application filed on November 30, 2005. *Id.* The abstract of Brams discloses:

Methods and systems for determining whether a component defined in a computer program is authorized to activate. One method can include receiving an activation authorization request from the component; accessing a license control file defining a plurality of mappings, each of the plurality of mappings associating at least one of a plurality of components defined in the computer program to at least one required license; determining if the component is associated with at least one required license based on the plurality of mappings; and providing a response to the component.

FEDEX-1006, Abstract.

57. Both Bram and the '416 Patent disclose a clustered computing system and techniques for its management. Bram discloses for example:

Embodiments of the invention provide *a method of determining whether a component defined in a computer program is authorized to activate*. The method can include receiving an activation authorization request from the component; accessing a license control file defining a plurality of mappings, each of the plurality of mappings associating at least one of a plurality of components defined in the computer program to at least one required license; *determining if the component is associated with at least one required license* based on the plurality of

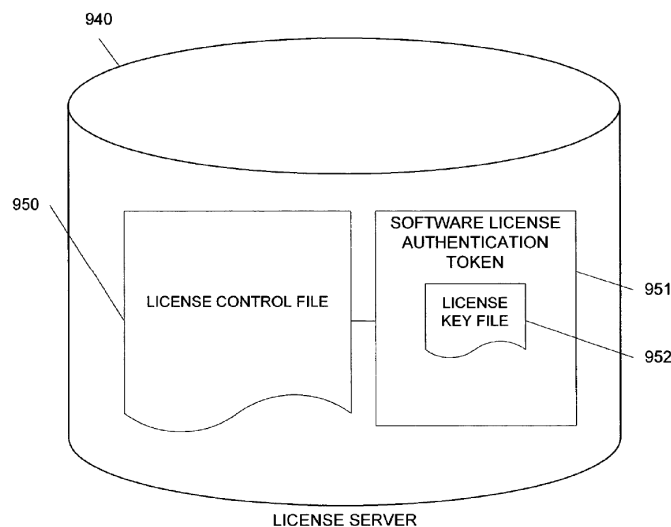
mappings; and providing a response to the component.

FEDEX-1006, [0004].

In some embodiments, the system 10 is designed to run on *one or more clusters of various types of computers, including heterogeneous clusters made up of different types of computers running different operating systems*. Each individual computer is called a “node.” *Multiple clusters can run simultaneously at multiple locations in order to provide mutual backup options for a single system*. As shown in FIG. 26, nodes 800 within a cluster 810 can communicate through multiple independent networks 820, such as local area networks (“LANs”), so that the failure of one network does not disrupt the operation of the cluster 810.

FEDEX-1006, [0232].

58. Bram describes a license server which includes a license control file and a software licensing authentication token. The token can include a license key file that defines the licenses available to a particular system user.



FEDEX-1006, Fig. 28. “FIG. 28 illustrates a license server including a license control file and a software licensing authentication token.” FEDEX-1006, [0029].

In some embodiments, system users obtain complete copies of the system 10 (i.e., software or code for all possible functionality provided with the system 10) and ***obtain licenses that allow them to run particular applications 16 or components of the system 10 (e.g., the applications 16 and components of the system 10 that the system user has purchased a license for)***. To manage and apply the licenses associated with a particular system user, the system 10 can include a license server. FIG. 28 illustrates a license server 940 according to one embodiment of the invention. As shown in FIG. 28, the license server 940 includes a license control file 950. The license control file 950 specifies what licenses are required in order to activate a particular application 16 or component of the system 10. The license control file 950 can be used to map activation authorization requests generated by applications 16 and/or components of the system 10 to licenses that are required in order for the license server 940 to authorize the activation of the applications 16 and/or components. In some embodiments, the license control file 950 includes one or more license files. For example, ***the license control file 950 can include a license file defining a base license or license package that allows core components of the system 10 to operate and one or more license files defining supplemental licenses that allow additional components of the system 10 to operate*** (e.g., ATM handling components, fraud management components, authorization components, etc.). In some embodiments, each license file can be associated with one or more licensable components of the system 10 and can define which components that when activated imply a use of the license, other components provided through a license, and/or other licenses and/or license packages associated with a license.

FEDEX-1006, [0244].

59. Bram further describes that the license server includes a software licensing authentication token (“token”) 951, and that the token includes a license key file 952 that defines the licenses available to a particular system user.

As shown in FIG. 28, the license server 940 also includes a software licensing authentication token (“token”) 951, which can include hardware and/or software. The license server 940 can log on to the token 951 (e.g., access and read data included in the token 951) and can manage access to the token 951. As shown in FIG. 28, ***the token 951 can include a license key file 952 that defines the licenses available to a particular system user. The license key file 952 provides authorization to apply the available licenses if constraints of the available licenses are satisfied by the current state of the system 10.*** Using the license key file 952, the license server 940 can authorize activation of applications 16 and/or components associated with available licenses and can disable (e.g., prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints. In some embodiments, ***each license key file 952 can specify parameters and/or constraints for each available license, such as an effective date, an expiration date, a total concurrent activations limit or count, a list of authorized named nodes, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., “restricted” or “unrestricted”) of components allowed under a license.***

FEDEX-1006, [0245].

C. The Sparks-Bram Combination

60. As I discuss below, the Sparks-Bram combination applies Bram’s teachings about licensing enforcement to Sparks’s distributed computing system. It would have been obvious to a POSITA to modify Sparks’s distributed computing system based on Bram’s disclosures on licensing such that the modified system applies and tracks licenses as appropriate for various system components to ensure compliance with license requirements. For example, it would have been obvious to

a POSITA to include the functionality provided by Bram's licensing server in the distributed computing system of Sparks. A POSITA would have been motivated to incorporate the teachings of Bram into Sparks's system with a reasonable expectation of success.

61. I note, as an initial matter, that Sparks and Bram are in the same field of endeavor. As I discuss above in Sections IX.A and IX.B, Sparks and Bram both describe clustered computing systems and their management. FEDEX-1005, 1:7-8, 1:33-39; FEDEX-1006, [0004], [0232]. It is my opinion that a POSITA would have been motivated to pursue the Sparks-Bram combination for multiple reasons.

62. A POSITA equipped with basic common sense and ordinary creativity would have sought out teachings from references like Bram to incorporate licensing support in Sparks's distributed computing system based on the teachings and suggestions within Sparks. As I discuss more in Section X.A, Sparks already automates configuration/enablement of tiers based on administrator-defined policies including requirements/constraints (e.g., usage, availability, min/max nodes in a tier, node/tier requirements, node identity). Adding Bram's license controls simply uses a known licensing mechanism to monitor requirements/constraints that Sparks already tracks (e.g., how many instances/nodes may run in a tier, node/tier requirements, node identity, etc.), yielding predictable enforcement of license compliance at cluster bring-up and runtime. In more detail, Sparks's system includes

“[o]ne or more control nodes [that] provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.” FEDEX-1005, 1:33-39. As discussed above, “[a] tier [within the model/system] is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission.” FEDEX-1005, 5:48–58. Sparks describes that “each node slot [of a tier] represents a data set that describes specific information for a corresponding node, such as *software resources* for a physical node that is assigned to the node slot.” FEDEX-1005, 6:22-30. Further, it was well-known before the Critical Date that software resources typically require a license. FEDEX-1007; FEDEX-1008. Moreover, license managers/servers were also well-known including those for cluster-based deployments. FEDEX-1011, p. 11 (“FLEXlm is the most popular license manager used in the software industry”); FEDEX-1012, p. 14 (discussing “Platform LSF” for “cluster administrators”), p. 231 (“[s]oftware licenses are valuable resources that must be fully utilized”; [t]his section discusses how LSF can help manage licensed applications to maximize utilization and minimize job failure due to license problems”). Sparks describes that a system administrator “may define policies regarding tier characteristics” as well as “specific node requirements for each tier of the fabric.” FEDEX-1005, 6:42-67. A

POSITA, in my opinion, would have understood that licensing is another common requirement to be configured for one or more nodes/tiers in Sparks.

63. It would have been obvious to a POSITA to incorporate in a computing system, such as the one of Sparks, some means to obtain the appropriate license(s) in order to operate a tier, including its nodes, in accordance with the prescribed license terms. Such means can include, for example, local license storage and/or a license server. FEDEX-1006, [0245] (“Using the license key file 952, the license server 940 can authorize activation of applications 16 and/or components associated with available licenses and can disable (e.g., prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints”), [0244] (“the system 10 can include a license server”).

64. Moreover, both references address clustered/distributed systems composed of “nodes.” *See* Sections IX.A, IX.B, and X.A. Bram targets clusters and defines license files, tokens, and key exchanges that gate component activation—exactly the type of control Sparks’s control node would host locally or via communications with an external server. *Id.* Further, enterprise deployments of clustered/distributed system typically require license compliance at scale. Bram provides concrete mechanisms (e.g., license key files, named nodes, OS/MAC/JVM based filters, concurrent activation counts, timers for re-checks) that align with enterprise policy enforcement workflows already modeled in Sparks. A POSITA

would naturally incorporate Bram's license checkpoints into Sparks's tier enablement steps.

65. Bram discloses additional benefits that would motivate the Sparks-Bram combination. For example:

Computer software providers often desire to control access to and activation of their computer software. In some embodiments, computer software provides create specialized versions of computer software for consumers that include only the functionality the computer software provider has authorized the consumers to perform. For example, *depending on functionality purchased, authorized, and/or needed by a specific consumer, the computer software provider can design specific computer software in order prevent the consumer from obtaining un-purchased, un-authorized, and/or un-needed functionality.*

FEDEX-1006, [0002].

Designing computer software, however, tailored to a specific purchase, authorization, and/or need of a consumer can be expensive and time-consuming. In an attempt to overcome the these deficiencies, some computer software providers create one version of computer software and provide consumer with licenses that authorize use of particular functionality provided by the computer system. Changing relationships between functionality of the system and the licenses required to activate the functionality, however, can be difficult to modify with modifying both the licenses and the computer software.

FEDEX-1006, [0003].

Licenses can also be tied to a system user's specific configuration so that system users cannot share licenses, and distributors cannot serve multiple system users with a single license. By customizing licenses

and separating licenses for individual components, licenses can be created that expire on various schedules and have different prices.

FEDEX-1006, [0247].

66. Given the benefits described by Bram and the knowledge of a POSITA, it would have been obvious to a POSITA to incorporate licensing capabilities in a distributed/clustered computing system to prevent the use of “un-purchased, unauthorized, and/or un-needed functionality.” FEDEX-1006, [0002]. It would have also been obvious to do so to “track applied licenses” within the system. FEDEX-1006, [0255] (“the license server 940 can use the floating license information to authorize or decline activation authorization requests generated by components of the system 10 and track applied licenses”), [0268] (“each component can include an identifier that keeps track of current license permissions granted by the license server 940”).

67. Further, it was well publicized that the unlicensed use of computer systems and/or programs may be unlawful or result in civil penalties. FEDEX-1007; FEDEX-1008. As such, it would have been obvious to a POSITA to include licensing capabilities in a computing system so that its users do not run afoul of laws or incur civil penalties. A POSITA in 2006 would have had multiple, concurrent reasons to integrate Bram’s license-enforcement mechanisms into Sparks’s distributed computing system. Enterprises deploying clustered software faced

material risks and costs from unlicensed use. Industry groups, such as the Business Software Alliance, widely warned that companies could be held liable for employees' unauthorized installations, face civil damages up to \$150,000 per program, and even criminal penalties up to \$250,000, and/or imprisonment for willful infringement. FEDEX-1008; FEDEX-1007. Integrating license based controls was therefore a straightforward, low-cost way (relative to the expensive fines/damages) to reduce legal exposure and ensure compliant use across clusters.

68. In addition to teachings and suggestions within Sparks and/or Bram regarding licensing, a POSITA would have been led to the Sparks-Bram combination because the licensing techniques that improved Bram's system in terms of license management and enforcement, would likewise be expected by the POSITA to improve the system of Sparks in the same way. Sparks's tiers are parameterized by admin policies (including minimum/maximum node counts). Bram teaches license constraints and counts (e.g., total concurrent activations) and lists of authorized named nodes — directly mappable as license-backed tier policies. A POSITA would have been motivated to account for license requirements, as taught by Bram, when implementing the technique Sparks teaches for setting up and activating the tiers of a distributed/clustered computing system. A POSITA would have realized that the same improvements offered by Bram's license management

and enforcement techniques in Bram's system would be realizable in other systems such as those disclosed by Sparks.

69. Further, a POSITA would have been led to the Sparks-Bram Combination because it involves the application of a known technique (Bram's licensing technique) to a known system (the clustered/distributed computing system of Sparks) to achieve predictable results (a clustered/distributed system having licensing capabilities) which address a widely-recognized business and compliance need, with a reasonable expectation of success. *See, e.g.,* FEDEX-1007; FEDEX-1008. Sparks already has policy and activation decisions in a control node (e.g., node minimums/maximums and tier admission policies among others), so adding Bram's standard license-server checks (e.g., named-node lists, activation counts, static/dynamic constraints) into the same enablement paths would have been a straightforward extension that yields predictable results: the same clustered fabric as Sparks, with license-gated activation and tracking as in Bram. FEDEX-1005, 6:22–30, 6:55–67, 9:22–27; FEDEX-1006, [0244]–[0245], [0263]–[0266], [0275]. Each element in combination merely performs the same function as it does separately.

70. **Reasonable Expectation of Success.** Given Sparks's disclosure of node and tier requirements (e.g., FEDEX-1005, 5:48–58, 6:22–30, 6:42–67), turning to a reference such as Bram to further incorporate licensing requirements would have

been straightforward from a POSITA's perspective. The Sparks-Bram combination represents the integration of known elements each performing the functions it had been known to perform to produce predictable results.

71. In fact, the Sparks-Bram combination does not change the key aspects of either reference. Both systems run on conventional nodes with processors and memory in clustered configurations. Bram's calls (which are used for get-key, constraint checks, etc.) are standard client/server interactions. Nothing in Sparks precludes such calls; indeed, Sparks's control node is designed to query/update a database and enforce policies during enablement — prime spots for Bram's checks. Moreover, Bram also contemplates multiple distributed license servers (primary/backup servers including geographically separated servers for disaster recovery) — a natural fit for Sparks' multi-tier/multi-domain fabrics. FEDEX-1006, [0232], [0252], [0304]. The crux of the combination simply involves adding functionality to Sparks's system based on the licensing disclosures of Bram. The respective teachings would work in combination just as they did apart. A POSITA would thus expect success integrating Bram's server calls with Sparks's control-node orchestration.

72. Further, integrating Bram's licensing techniques into the system of Sparks would have been a routine task for a POSITA. As I discuss above, Bram and Sparks are related to cluster computing systems and their management. FEDEX-

1005, 1:7-8, 1:33-39; FEDEX-1006, [0004], [0232]. Given their similarities, it would have been straightforward to modify the control node of Sparks to implement the functionality of and/or communicate with a licensing server such as the one disclosed by Bram. As such, a POSITA would have reasonably expected a successful outcome from combining Bram's teachings on licensing techniques and systems with Sparks's description of a distributed computing system.

X. GROUND 1: SPARKS AND BRAM

A. Claim 1

[1pre] A method of creating a clustered computing system, comprising:

73. It is my opinion that the Sparks-Bram combination renders [1pre] obvious even to the extent that the preamble is considered a limitation.

74. For example, Sparks describes methods for creating "a distributed computing system that conforms to a multi-level, hierarchical organization model" which "includes four distinct levels: fabric, domains, tiers and nodes."

In general, the invention is directed to a *distributed computing system that conforms to a multi-level, hierarchical organizational model*. One or more control nodes provide for the *efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model*.

As described herein, *the model includes four distinct levels: fabric,*

domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of the distributed computing system. The control nodes are responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

FEDEX-1005, 1:33-50. *See also id.*, 1:51-2:7.

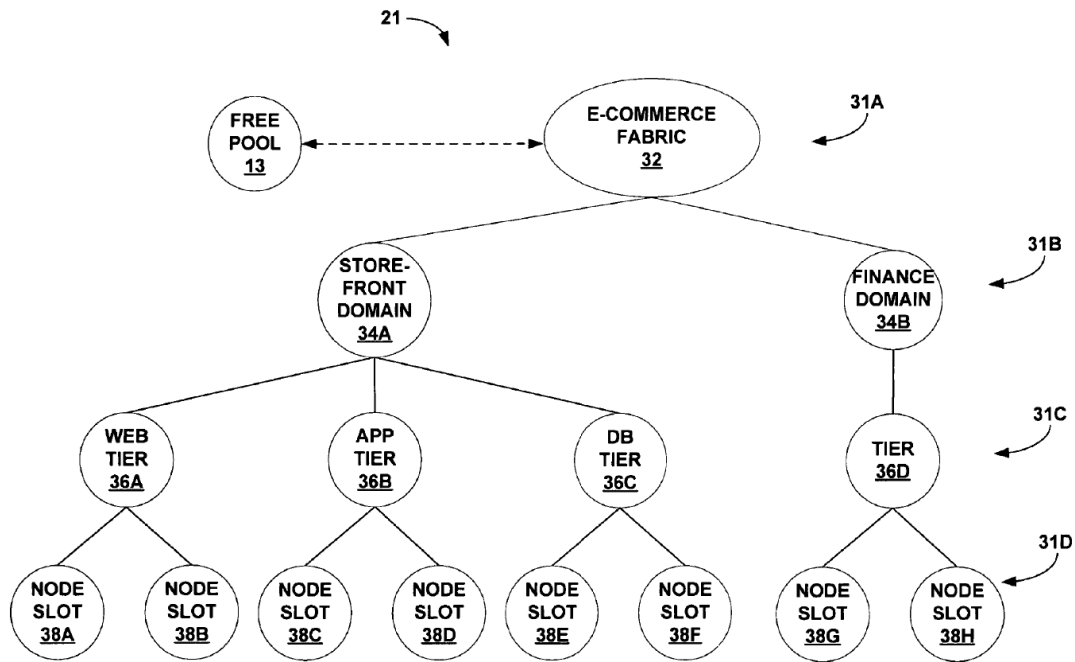
In general, distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

FEDEX-1005, 5:23–28.

A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

FEDEX-1005, 5:48–58.

75. An example of this model including its hierarchy is depicted in Fig. 2.



FEDEX-1005, Fig. 2; *see generally* 7:26-8:36. Sparks's distributed computing system includes computing nodes, and as I discuss below for [1a-i], [1a-ii], [1b], [1c], and [1d], Sparks's control node assigns computing nodes to the node slots within separate tiers.

FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. ***Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide high-performance processing.***

FEDEX-1005, 3:7-11; *see also id.*, Fig. 1.

76. A POSITA would have recognized that Sparks discloses, or at least renders obvious, a clustered computing system, because it was known to those of skill in the art that a distributed computing system is an example of a clustered

computing system. A POSITA would have understood that the computing nodes that make up a clustered computing system can be “distributed.” See also FEDEX-1009 (see definitions of distributed system, distributed processing, cluster, and clustering); FEDEX-1010 (see definitions of distributed, distributed processing, distributed system, cluster). Also, as would have been apparent, or obvious, to those of skill in the art that each tier (including its nodes) in Sparks is likewise itself a computing cluster within Sparks’s distributed computing system. This analysis is further consistent with the ’416 Patent’s description that “[t]he term ‘clustered computing system’ refers to a computing system formed from a group of loosely coupled computing devices that cooperate to provide the appearance that the devices function as a single system.” FEDEX-1001, 1:11-22; *see also id.*, 4:39-5:8, Fig. 2.

77. Bram similarly describes that a system can be “designed to run on one or more clusters of various types of computers.”

In some embodiments, the system 10 is designed to *run on one or more clusters of various types of computers, including heterogeneous clusters made up of different types of computers running different operating systems*. Each individual computer is called a “*node*.” Multiple clusters can run simultaneously at multiple locations in order to provide mutual backup options for a single system. As shown in FIG. 26, nodes 800 within a cluster 810 can communicate through multiple independent networks 820, such as local area networks (“LANs”), so that the failure of one network does not disrupt the operation of the cluster 810.

FEDEX-1006, [0232]. Bram's confirmatory teachings reinforce how a POSITA would have understood Sparks's above-discussed similar disclosures regarding a collection of coordinated computing nodes making up a distributed and clustered computing system. FEDEX-1006, [0232], [0252], [0304]. This leaves no doubt that the Sparks-Bram combination provides the claimed limitation of "creating a clustered computing system."

[1a-i] storing license information for a computing cluster in a memory module associated with a computing node,

78. It is my opinion that the Sparks-Bram combination renders [1a] obvious.

79. Sparks describes a control node that executes administration software to logically define the hierarchical organization of a distributed computing system, where the software creates "a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric."

System administrator 20 accesses administration software executing on control node 12 *to logically define the hierarchical organization of distributed computing system* 10. For example, system administrator 20 may provide input to develop an organizational model 21 for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, *develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.*

FEDEX-1005, 6:1-9.

80. Sparks further describes a database that stores information pertaining to the distributed computing system including, among other things, its tiers, and node slots of each tier.

Next, control node 12 retrieves the node requirements associated with the selected tier (62). ***Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each of the tiers.*** Control node 12 retrieves the node requirements for the selected tier from the database.

FEDEX-1005, 9:22-27.

In one embodiment, a method comprises detecting a node added to a network and automatically identifying attributes for the detected node. The method further comprises ***updating a model*** to store the attributes for the detected node, wherein the model defines an organization of a distributed computing system having a plurality of computing nodes. The method further comprises automatically deploying the detected node within the distributed computing system in accordance with the model based on the attributes.

FEDEX-1005, 1:51-59.

A computer-implemented method comprising: ***storing a model within a database of a control node***, wherein the model defines an organization of a distributed computing system having a plurality of computing nodes...

FEDEX-1005, 23:15-20.

81. In my opinion, a POSITA would have understood that Sparks discloses or at least renders obvious storing information for a computing cluster in a memory

module associated with a computing node. A POSITA would have recognized that Sparks discloses, or at least renders obvious, a computing cluster, because it was known to those of skill in the art that a distributed computing system is an example of a computing cluster. FEDEX-1009 (see definitions of distributed system, distributed processing, cluster, and clustering); FEDEX-1010 (see definitions of distributed, distributed processing, distributed system, cluster). A POSITA would have understood that the computing nodes that make up a clustered computing system can be “distributed.” Also, as would have been apparent, or obvious, to those of skill in the art that each tier (including its nodes) in Sparks is likewise itself a computing cluster within Sparks’s distributed computing system. My discussion above for [1pre] is also relevant here.

82. Further, a computing node in Sparks such as a control node or an application node, includes a memory module to store information.

FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. ***Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide high-performance processing.***

In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated pool 15 and a maintenance pool 17. In addition, ***distributed computing system 10 includes at least one control node 12.***

Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

FEDEX-1005, 3:7-25.

In general, allocated pool 15 includes application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. ***Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory).*** Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated pool 15.

FEDEX-1005, 3:49-59. *See also id.*, Fig. 1.

83. A POSITA would have understood or it would have been obvious to a POSITA that information for Sparks's distributed computing system including its model and requirements for each tier and each node thereof is stored by the control node in a memory module. For example, the control node's memory can store some or all of Sparks's database. Sparks teaches that a control node's rule engine may correspond to a tier, and that it stores read-only expected state information (e.g., max nodes) in its working memory. FEDEX-1005, 16:36-40, 17:1-7, Figs. 17-18. A

POSITA would have understood Sparks to teach, or would have found it obvious, that the working memory of a control node's rule engine corresponding to a tier would store the licensing information as part of its expected state in the Sparks-Bram combination. Alternatively, a POSITA would have understood or it would have been obvious to a POSITA that information retrieved from the database can also be stored in a memory module associated with a computing node such as a control node or an application node.

84. Alternatively, and additionally, a POSITA would have understood Sparks to teach, or would have found obvious in view of Sparks, that the licensing information would have been stored in the BLT's goal manager 312. Sparks teaches that an administrator defines a goal for the expected state of the fabric, and this goal is managed by the goal manager. FEDEX-1005, 17:65-18:2. Sparks also teaches that it "may automatically generate a goal of harvesting a target number of nodes to enable the tier." *Id.*, 18:47-50. Sparks teaches that "administrator 20 may interact with BLT 206 to define an expected state 210 for the fabric. BLT 206 communicates expected state 210 to SLAI 204." *Id.*, 14:11-14. Thus, a POSITA would understand that licensing requirements in the Sparks-Bram combination would be stored in both the BLT's goal manager, and the expected state portion of the working memory of an application rules engine associated with the tier.

85. Bram's system can be "designed to run on *one or more clusters of various types of computers*." FEDEX-1006, [0232]. Bram's system users "obtain licenses that allow them to run particular applications 16 or components of the system 10." *Id.*, [0244]. Bram further discloses:

As shown in FIG. 28, the *license server 940 also includes a software licensing authentication token ("token") 951*, which can include hardware and/or software. The license server 940 can log on to the token 951 (e.g., access and read data included in the token 951) and can manage access to the token 951. As shown in FIG. 28, *the token 951 can include a license key file 952 that defines the licenses available to a particular system user. The license key file 952 provides authorization to apply the available licenses if constraints of the available licenses are satisfied by the current state of the system 10.* Using the license key file 952, the *license server 940 can authorize activation of applications 16 and/or components associated with available licenses* and can disable (e.g., prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints. In some embodiments, *each license key file 952 can specify parameters and/or constraints for each available license*, such as an effective date, an expiration date, a total concurrent activations limit or count, a list of authorized named nodes, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., "restricted" or "unrestricted") of components allowed under a license.

FEDEX-1006, [0245]. Bram's system components can also obtain and store updated licensing information such as "updated license control file 950, an updated token 951, and/or an updated license key file 952." FEDEX-1006, [0282]. Further, Bram discloses obtaining and storing licensing information such a license control file (950), license key file, within the token (951, 952) are expressly obtained and stored

so that the system can evaluate license constraints before authorizing activation; Bram also teaches updating/refreshing the license server information (e.g., updated control file, token, and key file), reinforcing storage at the node/cluster side to enable checks. FEDEX-1006, [0232], [0244], [0245], [0282], [0005], Fig. 28.

86. In addition to the reasons I provided in Section IX.C, a POSITA would have been motivated to account for license requirements, as taught by Bram, when implementing the techniques Sparks teaches for setting up and activating one or more tiers of a distributed computing system. It would have been obvious to a POSITA to store license information, such as that disclosed by Bram, as part of the above-discussed information stored in the memory module of a node for a tier (computing cluster) in Sparks so that each node and/or the number of nodes of a tier comport with license requirements. I provide additional details in my discussion for [1a-ii] regarding the stored information, including license information.

[1a-ii] 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;

87. It is my opinion that the Sparks-Bram combination renders [1a] obvious.

88. The Sparks-Bram combination provides licensing information as I discuss above for [1a-i]. Note that a POSITA would have understood that “the licensing information” in [1a-ii] refers to “license information” in [1a-i].

89. Sparks further describes information that includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster. As I discuss above for [1pre] and [1a-i], Sparks describes a computing node, called a control node, that provides a user interface to define properties of a tier (computing cluster).

System administrator 20 accesses administration software executing on control node 12 to ***logically define the hierarchical organization of distributed computing system 10***. For example, system administrator 20 may provide input to develop an organizational model 21 for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.

FEDEX-1005, 6:1-9.

System administrator 20 may further ***define specific node requirements for each tier of the fabric***. For example, the node requirements specified by system administrator 20 may include a ***central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics*** that may be detected on the individual computing nodes. System administrator 20 may also specify user-defined hardware attributes of the computing nodes, such as whether I/O (like HBA) is required. The user-defined hardware attributes are typically not capable of detection during an automatic inventory. In this manner,

system administrator 20 creates a list of attributes that the tier requires of its candidate computing nodes.

FEDEX-1005, 6:42-54.

Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each of the tiers. Control node 12 *retrieves the node requirements for the selected tier* from the database.

FEDEX-1005, 9:22-27. *See also id.*, 7:1-14 (“control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20”), 11:50–12:5, 12:19-24, Figs. 7-8.

90. One of the tier properties *identifies a size of the tier (cluster)*. Sparks discloses for example:

- “System administrator 20 may define policies regarding tier characteristics, such as a *minimum number of nodes a tier requires*, ...” (FEDEX-1005, 6:55-67),
- “For each of the tiers, control node 12 *creates a number of “node slots” equal to the maximum number of application nodes 14 that may be deployed*. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot.” (6:22-30),
- “Control node 12 then determines whether *the node count for the tier is met* (74). Control node 12 may, for example, determine *whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities*. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20.” (9:52-62),
- Control node 12 determines whether the *minimum node count for the tier* is

met (82).” (10:15-35).

91. In my opinion, a POSITA would have understood that a tier property such as a “minimum number of nodes” (or “maximum number”) is a ***bundle-type parameter*** because this number defines a size requirement for a ***tier (cluster)***. This analysis is consistent with the ’416 Patent which discloses:

The licensing scheme may implement a ***multi-tiered (or bundled)*** license program. The first part of the license scheme defines a permissible size range for a cluster. In one embodiment, ***clusters may be categorized into multiple size ranges, referred to herein as bundle-types***. In alternate embodiments, 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. A bundle may be sold as a set or group of node or grid types to meet a specific market or function. For example, a product category may include one or more bundle sizes that served a single purpose such as, e.g., array/LUN purpose. Bundles may have a specific homogeneous purpose along with standard grid parameters such as, e.g., right to use, ***maximum and minimum node counts***, and a specific array type functional brand.

FEDEX-1001, 5:48-64.

92. Another tier property identifies ***at least one additional bundle-type characteristic of the tier (cluster)***. Sparks discloses that tier properties include “a name for the tier, a description of the tier, and a image associated with the tier.” FEDEX-1005, 11:66-12:3. Additional node requirements for a tier can include “a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics.” FEDEX-1005,

6:42–54. Tier properties can include a common functional aspect or attribute such as “usage, availability model or business service mission.”

A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

FEDEX-1005, 5:48–58.

More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation management (CRM) service. System administrator 20 further defines *one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain*. As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a web tier, an application tier and a database tier. In this manner, distributed computing system 10 may be configured to automatically *provide web server functions, business application functions and database functions*.

FEDEX-1005, 6:10-21.

93. A POSITA would have understood that these additional properties (taken individually or collectively) provide a *bundle-type characteristic* because each additional property defines an operative aspect for a *tier (cluster)*.

94. This analysis is consistent with the '416 Patent which discloses:

The licensing scheme may implement a *multi-tiered (or bundled)* license program. The first part of the license scheme defines a permissible size range for a cluster. In one embodiment, *clusters may be categorized into multiple size ranges, referred to herein as bundle-types*. In alternate embodiments, 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. *A bundle may be sold as a set or group of node or grid types to meet a specific market or function*. For example, a product category may include one or more bundle sizes that served a single purpose such as, e.g., array/LUN purpose. *Bundles may have a* specific homogeneous purpose along with standard grid parameters such as, e.g., right to use, *maximum and minimum node counts*, and a specific array type functional brand.

FEDEX-1001, 5:48-64.

95. Bram discloses that the licensing information includes the one or more node license parameters that identifies a characteristic of a computing node. Bram describes a system “designed to run on one or more clusters of various types of computers.” FEDEX-1006, [0232]. Bram further discloses:

In some embodiments, *system users obtain complete copies of the system 10 (i.e., software or code for all possible functionality provided with the system 10) and obtain licenses that allow them to run particular applications 16 or components of the system 10 (e.g., the applications 16 and components of the system 10 that the system user has purchased a license for). To manage and apply the licenses associated with a particular system user, the system 10 can include a license server.* FIG. 28 illustrates a license server 940 according to one embodiment of the invention. As shown in FIG. 28, the license server 940 includes a license control file 950. *The license control file 950 specifies what licenses are required in order to activate a particular application 16 or component of the system 10.* The license control file 950 can be used to map activation authorization requests generated by applications 16 and/or components of the system 10 to licenses that are required in order for the license server 940 to authorize the activation of the applications 16 and/or components. In some embodiments, the license control file 950 includes one or more license files. For example, *the license control file 950 can include a license file defining a base license or license package that allows core components of the system 10 to operate and one or more license files defining supplemental licenses that allow additional components of the system 10 to operate* (e.g., ATM handling components, fraud management components, authorization components, etc.). In some embodiments, each license file can be associated with one or more licensable components of the system 10 and can define which components that when activated imply a use of the license, other components provided through a license, and/or other licenses and/or license packages associated with a license.

FEDEX-1006, [0244].

96. The license server further includes a “software licensing authentication token” with a “license key file” that “provides authorization to apply the available licenses if constraints of the available licenses are satisfied” and further specifies parameters and/or constraints for each available license.

As shown in FIG. 28, the license server 940 also includes a software licensing authentication token (“token”) 951, which can include hardware and/or software. The license server 940 can log on to the token 951 (e.g., access and read data included in the token 951) and can manage access to the token 951. As shown in FIG. 28, the token 951 can include a license key file 952 that defines the licenses available to a particular system user. ***The license key file 952 provides authorization to apply the available licenses if constraints of the available licenses are satisfied by the current state of the system 10.*** Using the license key file 952, the license server 940 can authorize activation of applications 16 and/or components associated with available licenses and can disable (e.g., prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints. In some embodiments, ***each license key file 952 can specify parameters and/or constraints for each available license, such as an effective date, an expiration date, a total concurrent activations limit or count, a list of authorized named nodes, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., “restricted” or “unrestricted”) of components allowed under a license.***

FEDEX-1006, [0245]. Bram further discloses that “the license server 940 can check static constraints (e.g., host restrictions) and dynamic constraints (e.g., counts of the number of instances of a component in the system 10 and/or on a specific node) for a required license associated with a component.” *Id.*, [0275].

97. Bram includes several figures (Figs. 28A-D) showing examples of a license key file.

952a

LICENSE KEY FILE	
LICENSE NAME:	CORE
LICENSE MODE:	FULL (UNRESTRICTED)
LICENSE AUTHORIZED HOST(S):	NODE1, NODE2
LICENSE EFFECTIVE DATE:	N/A
LICENSE EXPIRATION DATE:	N/A
EFFECTS:	
	-CORE COMPONENTS ACTIVATED IN UNRESTRICTED MODE
	-CORE COMPONENTS ONLY ALLOWED TO ACTIVATE ON ON NODE1 AND NODE2
	-ATTEMPTS TO ACTIVATE CORE COMPONENTS ON NODES OTHER THAN NODE1 AND NODE2 REJECTED
	-CONFIGURED NON-CORE COMPONENTS ACTIVATED IN UNRESTRICTED MODE
	-AUTHORIZATION COMPONENTS NOT ALLOWED TO ACTIVATE
LICENSE NAME:	FRAUD MANAGER
LICENSE MODE:	EVALUATION (RESTRICTED)
LICENSE AUTHORIZED HOST(S):	NODE1, NODE2
LICENSE EFFECTIVE DATE:	FIRST ACTIVATION
LICENSE EXPIRATION DATE:	60 DAYS FROM FIRST ACTIVATION
LICENSED COPIES:	1
EFFECTS:	
	-FRAUD MANAGEMENT COMPONENTS ACTIVATED IN RESTRICTED MODE
	-FRAUD MANAGEMENT COMPONENTS ONLY ALLOWED TO ACTIVATE ON NODE1 AND NODE2
	-ON FIRST ACTIVATION OF FRAUD MANAGEMENT COMPONENTS EXPIRATION DATE SET IN TOKEN
	-FRAUD MANAGEMENT COMPONENTS VALID FOR 60 DAYS FROM FIRST ACTIVATION
	-ATTEMPTS TO ACTIVATE FRAUD MANAGEMENT COMPONENTS AFTER EXPIRATION DATE REJECTED
	-ATTEMPTS TO ACTIVATE FRAUD MANAGEMENT COMPONENTS ON NODES OTHER THAN NODE1 AND NODE 2 REJECTED
	-ONE FRAUD MANAGEMENT COMPONENT (ONE COPY) ALLOWED TO BE ACTIVATE AT ONE TIME
LICENSE NAME:	ATM HANDLING
LICENSE MODE:	EVALUATION (RESTRICTED)
LICENSE AUTHORIZED HOST(S):	NODE1
LICENSE EFFECTIVE DATE:	N/A
LICENSE EXPIRATION DATE:	N/A
EFFECTS:	
	-ATM HANDLING COMPONENTS ACTIVATED IN RESTRICTED MODE
	-ATM HANDLING COMPONENTS ONLY ALLOWED TO ACTIVATE ON NODE1
	-ATTEMPTS TO ACTIVATE ATM HANDLING COMPONENTS ON NODES OTHER THAN NODE1 REJECTED

FEDEX-1006, Fig. 28C. *See also id.*, [0030], [0245].

98. Bram also discloses that a client, to activate license covered components, broadcasts a get-key request in order to obtain a decryption key from a

license server (conditioned on a license check) to decrypt encrypted files (encrypted to prevent unlicensed used). Bram discloses for example:

- “When an application 16 of the system is initialized, the application 16 (e.g., the application licenser component) can broadcast a *get-key request*.” (FEDEX-1006, [0263]),
- “Next, *the license server 940 can check that the application 16 that generated the get-key request meets the license constraints of the basic license* (step 1006). For example, the license server 940 can *verify that the node executing the application 16 that generated the get-key request is a valid node*.” ([0265]),
- “After authenticating the get-key request, the license server 940 can obtain a decryption key from the token 951 (step 1008). If the license server 940 successfully obtains the decryption key from the token 951 (step 1010), the license server 940 can format, sign, and send a response, which includes the decryption key, to the application 16 that submitted the get-key request (step 1012).” ([0266]),
- “For example, the license server 940 can check *static constraints (e.g., host restrictions)* and dynamic constraints (e.g., counts of the number of instances of a component in the system 10 and/or on a specific node) for a required license associated with a component.” ([0275]),
- “In some embodiments, each license key file 952 can specify *parameters and/or constraints for each available license*, such as an effective date, an expiration date, a total concurrent activations limit or count, *a list of authorized named nodes*, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., “restricted” or “unrestricted”) of components allowed under a license.” ([0245]).

99. In my opinion, a POSITA would have understood Bram to teach the claimed one or more node license parameters that identify a characteristic of a computing node because such parameters are checked by a license server in response to receiving a get-key request. Examples of such parameters include “a list of authorized named nodes, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., ‘restricted’ or ‘unrestricted’) of components allowed under a license.” FEDEX-1006, [0245]. Indeed, Bram describes that a node name/identity of a get-key request can be matched against “a list of authorized named nodes” ([0245]) for example, to determine whether the node is “valid” ([0265]).

[1b] initializing the computing cluster in a first computing node;

100. It is my opinion that the Sparks-Bram combination renders [1b] obvious. Sparks describes various computing cluster initialization steps.

101. Sparks for example describes initializing the computing cluster in a first computing node such as the control node. Sparks’s control node is a type of computing node in Sparks.

FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. ***Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide high-performance processing.***

In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated pool 15 and a maintenance pool 17. In addition, ***distributed computing system 10 includes at least one control node 12.***

Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

FEDEX-1005, 3:7-25.

In general, allocated pool 15 includes application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. ***Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory).*** Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated pool 15.

FEDEX-1005, 3:49-59.

102. Sparks discloses that the control node can be used to define the hierarchical organization of Sparks's distributed computing system, which includes domains, tiers, and node slots. The control node, for each tier, creates a number of node slots equal to the maximum number of application nodes.

System administrator 20 accesses administration software executing on control node 12 ***to logically define the hierarchical organization of distributed computing system 10.*** For example, system administrator 20 may provide input to develop an organizational model 21 for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, ***develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.***

FEDEX-1005, 6:1-9.

For each of the tiers, control node 12 creates a number of “node slots” equal to the maximum number of application nodes 14 that may be deployed. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot. The node slots may, for instance, identify a particular software image instance associated with an application node 14 as well as a network address associated with that particular image instance.

FEDEX-1005, 6:22-31. As I discuss above, a tier (including its computing nodes) is an example of a computing cluster. *See* Sections IX.A; X.A [1pre], [1a-i], [1a-ii].

103. Moreover, the control node “maintain[s] a database having entries for each node slot, where the entries identify the node requirement for each of the tiers” and “retrieves the node requirements for the selected tier from the database.”

Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each of the tiers. Control node 12 retrieves the node requirements for the selected tier from the database.

FEDEX-1005, 9:22-27.

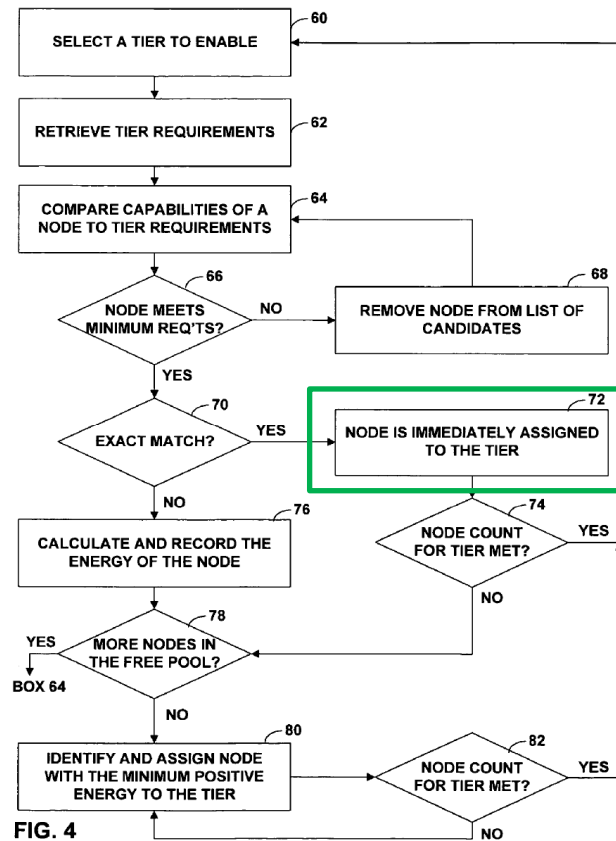
104. A POSITA would have recognized that operations such as defining a tier (computing cluster), creating the node slots of a tier, and maintaining/retrieving a database entry for each of the nodes in a tier satisfy the claimed “initializing” step because of their role in the sequence of steps that leads to an activated computing cluster (see my discussion below for [1c] and [1d]). These operations can be viewed as creating (initializing) the “blueprints” from which the computing cluster is initialized. For example, the tier’s node slots are required to be created (initialized) before computing nodes called application nodes are assigned to them. Moreover, these operations are performed in Sparks’s control node. FEDEX-1005, 1:33-39, 6:1–9, 6:22–31, 9:22–27.

105. Alternatively, it is my opinion that Sparks’s assignment of a computing node to a node slot of a tier also satisfies [1b]. Based on checking a computing node against the node requirements of the tier, Sparks’s control node can assign a computing node from the free pool to a node slot of the tier. FEDEX-1005, 9:43-51 (discussing that “[i]f the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72)”).

If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the *node*

requirements of the tier (70). If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).

FEDEX-1005, 9:43-51.



FEDEX-1005, Fig. 4 (annotated). *See also id.*, 9:3-13, 9:14-21, 9:52-62, 10:15-26, 10:27-34, Fig. 3.

106. Indeed, a POSITA would have recognized that Sparks discloses initializing the computing cluster in a first computing node based on Sparks's assignment of a first computing node to a tier (computing cluster). As shown above,

Sparks's step 72 assigns a node that meets minimum tier requirements to the tier. This analysis is further consistent with the '416 Patent which discloses that "the cluster may be initialized with as few as one member."

If, at operation 520, the node that initiated the license request does not have a node license of its own and there are licenses left in the node license pool, then one of the node licenses may be conferred to the node receiving the management command. Operation 525 may be done in two phases. In the first phase the node license pool may be updated to associate a unique identifier (e.g., a serial number) with the node that is assigned the license. In the second phase, the node may be "branded" as a licensed node having a specific bundle type, e.g., by writing a token to a protected non-volatile memory location on the node. The node license may then be assigned specifically to the node (and may not be assigned to any other node). Control may then pass to operation 530, described below.

FEDEX-1001, 7:19-32.

By contrast, if 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).

FEDEX-1001, 7:33-41.

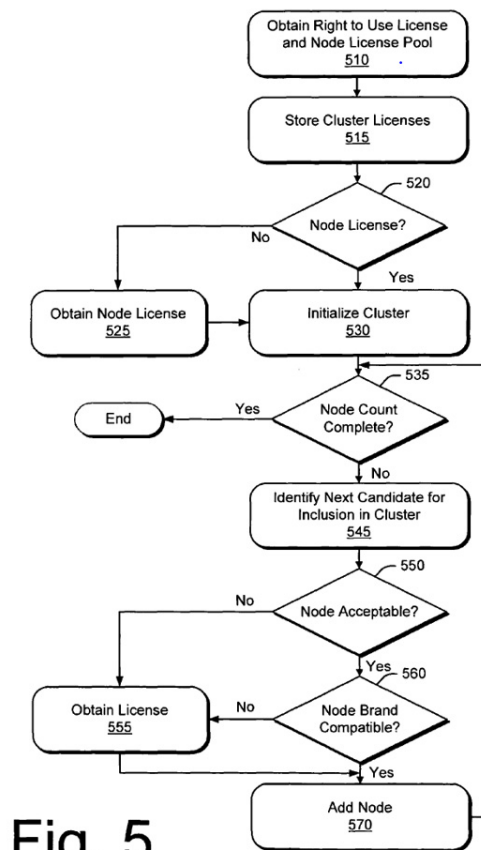


Fig. 5

FEDEX-1001, Fig. 5 (see step 530, “Initialize Cluster”).

**[1c] adding one or more available computing nodes to the computing cluster;
and**

107. It is my opinion that the Sparks-Bram combination renders [1c] obvious.

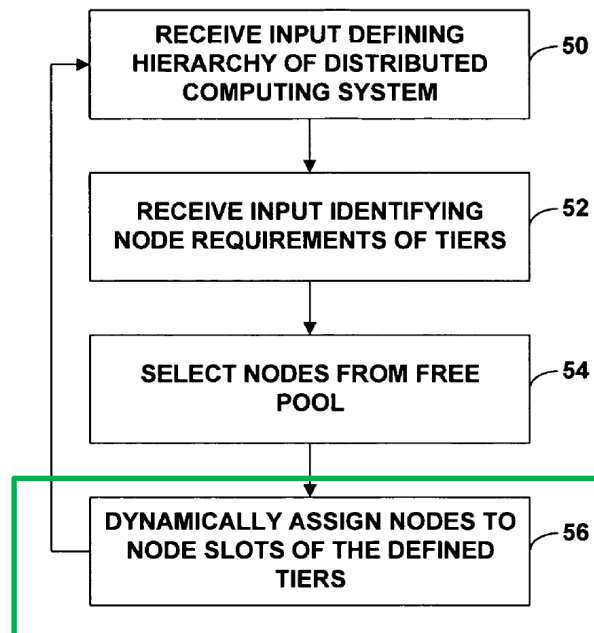
108. The control node of Sparks populates the node slots of a tier (computing cluster), e.g., “*assigns computing nodes to the node slots of each tier* based on the node requirements specified for the tiers and the identified node attributes.”

Control node 12 dynamically *assigns computing nodes to the node*

slots of each tier based on the node requirements specified for the tiers and the identified node attributes (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a “best fit” algorithm.

FEDEX-1005, 9:3-13. This assignment step (56) is part of Fig. 3 which shows the “operation of control node 12 when configuring distributed computing system 10.”

FEDEX-1005, 8:37-39.



FEDEX-1005, Fig. 3 (annotated).

109. Spark further describes that a control node adds an available computing node to a tier (computing cluster). Sparks refers to an available computing node as an unallocated node or a node from free pool.

- “Control node 12 automatically ***assigns unallocated nodes from free pool 13 to respective tier node slots of a tier.***” (FEDEX-1005, 7:1-14),
- “Control node 12 may automatically ***add unallocated nodes*** from free pool 13 to a tier.” (7:15-25),
- “Additionally, control node 12 may also ***add nodes from free pool 13 to a tier when more processing capacity is needed within the tier***, remove nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.” (8:14-25),
- “Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated pool 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to ***add an application node to allocated pool 15*** to increase processing capacity of distributed computing system 10.” (3:40-48). *See also id.*, 8:61-9:2, 9:28-30.

110. Sparks provides additional details of node assignment which are described in reference to Figure 4. Based on checking a computing node against the node requirements of the tier, the control node can assign a computing node from the free pool to a node slot of the tier.

FIG. 4 is a flow diagram illustrating exemplary ***operation of control***

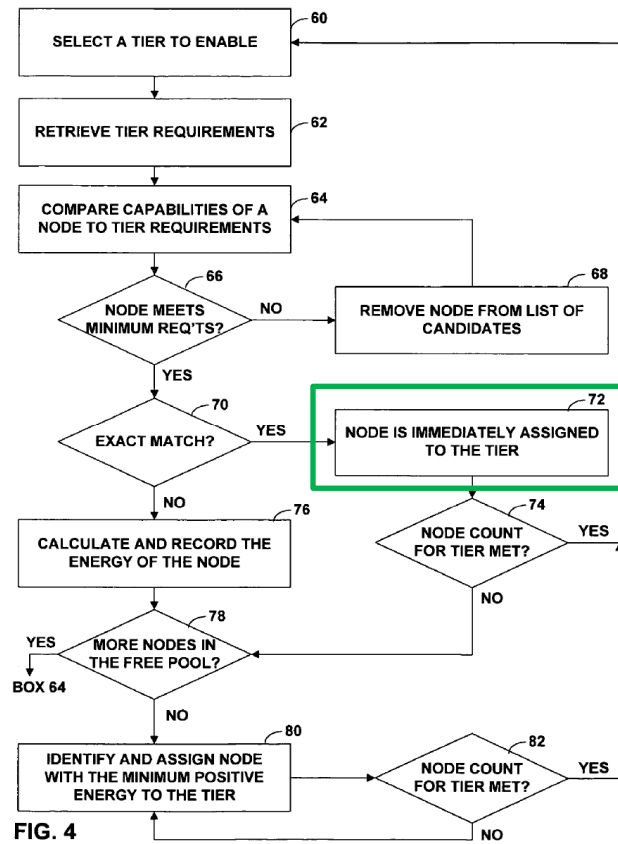
node 12 when assigning computing nodes to node slots of tiers.

Initially, control node 12 selects a tier to enable (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

FEDEX-1005, 9:14-21.

If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the node requirements of the tier (70). ***If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).***

FEDEX-1005, 9:43-51.



FEDEX-1005, Fig. 4 (annotated). *See also id.*, 9:52-62, 10:15-26, 10:27-34.

111. In my opinion, a POSITA would have recognized that Sparks discloses adding (see discussion above for assigning steps in Figures 3 and 4) one or more available computing nodes (“unallocated nodes that are available for use”) to the computing cluster (tier in Sparks’s distributed computing system).

[1d] activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.

112. It is my opinion that the Sparks-Bram combination renders [1d] obvious.

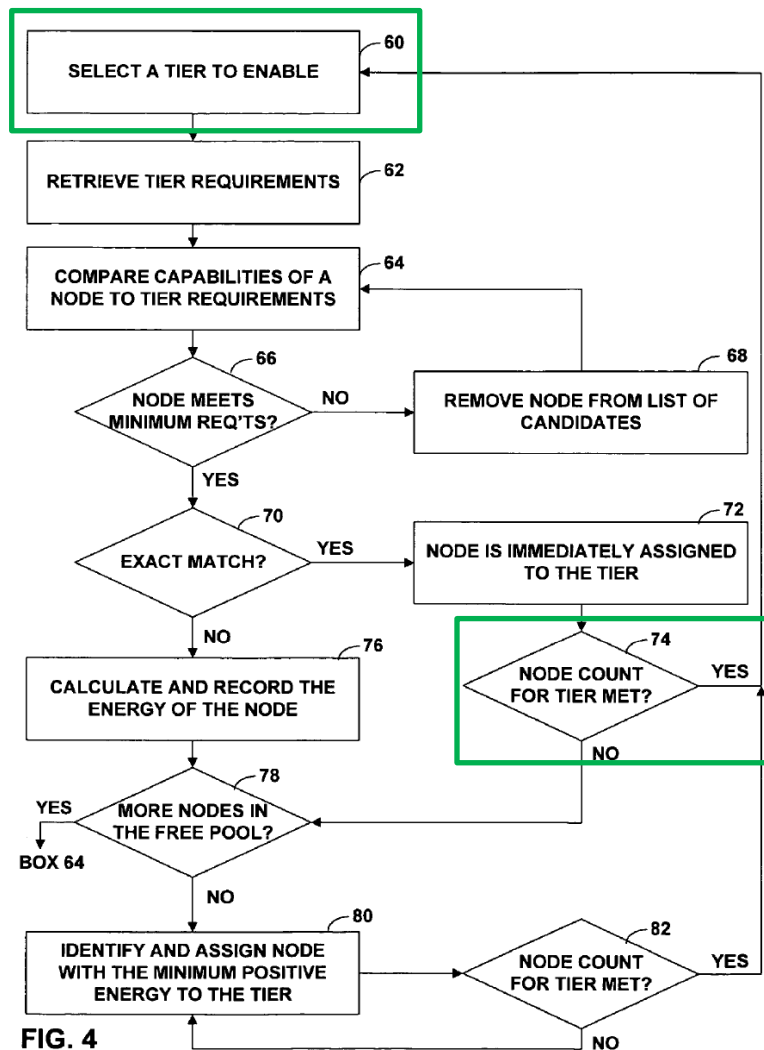
113. Sparks describes “operation of control node 12 when assigning computing nodes to node slots of tiers,” which is shown in Fig. 4.

FIG. 4 is a flow diagram illustrating exemplary *operation of control node 12* when assigning computing nodes to node slots of tiers. Initially, *control node 12 selects a tier to enable* (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

FEDEX-1005, 9:14-21.

Control node 12 then *determines whether the node count for the tier is met* (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20. *When the node count for the tier is met, control node 12 selects the next tier to enable*, e.g., the tier with the next largest priority, and *repeats the process until all defined tiers are enabled*, i.e., populated with application nodes (60).

FEDEX-1005, 9:52-62.



FEDEX-1005, Fig. 4 (annotated). Sparks additionally discloses that “a user may request that a newly created tier come online” via enabling the tier.

Goal manager 312 may automatically formulate dependencies between goals or may allow a user to specify the dependencies. For example, a user may request that a newly created tier come online, and goal manager 312 may automatically generate a goal of harvesting a target number of nodes to enable the tier.

FEDEX-1005, 18:45–50.

114. A POSITA would have recognized that Sparks discloses activating the computing cluster (e.g., “enable” a tier) when the computing cluster includes a number of nodes that complies with the bundle-type parameter (e.g., “[w]hen the node count for the tier is met”). Moreover, a POSITA would have understood or it would have been obvious to a POSITA that the computing cluster is activated when for example, the node count for the tier is met because there would be enough nodes in a tier to perform the tier’s assigned function. This analysis is further consistent with how a cluster is activated in the specification of the ’416 Patent:

If, at operation 535, the node count for the cluster has reached a limit specified in the Right-to-Use license (or if there are no more candidate nodes), the cluster formation process is complete and the process ends. By contrast, if at operation 535 *the node count for the cluster has not been reached*] (i.e., *if the cluster has not yet activated a number of nodes authorized by the license terms*), then control passes to operation 545 and *a next candidate node for inclusion in the cluster may be identified*. This can be done using any number of techniques common in the art.

FEDEX-1001, 7:42-51.

115. A POSITA would have also understood that compliance with the at least one additional bundle-type characteristic of the cluster is evidenced through Sparks’s disclosure of requirement checks before a node is added to the tier (cluster). Sparks discloses, for example, that “control node 12 determines whether the node attributes of the computing node meets the minimum node requirements of the tier

(66)” and if they do, the computing node is assigned to a node slot of the tier. FEDEX-1005, 9:22-51.

116. Moreover, Bram also activates components if constraints are satisfied. Bram requires, for example, that license constraints are satisfied before activation and uses counts among its dynamic constraints and concurrent activation limits. FEDEX-1006, [0244] (“The license control file 950 specifies what licenses are required in order to activate a particular application 16 or component of the system 10”), [0245]. A POSITA would have additionally understood that using Bram’s license check at Sparks’s enablement transition is common sense: e.g., enable only when $\text{count} \geq \text{min}$ and license constraints are satisfied.

B. Claim 2

[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.

117. It is my opinion that the Sparks-Bram combination renders [2] obvious.

118. The Sparks-Bram combination renders obvious storing license information for a computing cluster in a memory module associated with a computing node for at least the reasons I discuss above for [1a-i] and [1a-ii]. It would have been apparent or at least obvious to a POSITA that Bram’s teachings, as

integrated with Sparks, provides retrieving licensing information from an external licensing server for the following reasons.

119. Bram discloses retrieving licensing information from an external licensing server. Bram describes for example:

To manage and apply the licenses associated with a particular system user, the system 10 can include a *license server*.

FEDEX-1006, [0244].

Using the license key file 952, the *license server* 940 can authorize activation of applications 16 and/or components associated with available licenses and can disable (e.g., prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints.

FEDEX-1006, [0245].

Another embodiment provides a system for determining whether a component defined in a computer program is authorized to activate. The system can include a license control file defining a plurality of mappings, each of the plurality of mappings associating at least one of a plurality of components defined in the computer program with at least one required license, and a *license server* configured to receive an activation authorization request from the component, to determine if the component is associated with at least one required license based on the plurality of mappings, and to provide a response to the component.

FEDEX-1006, [0005]. *See also id.*, [0006].

120. A POSITA would have recognized or it would have been obvious that Bram's *license server* is an external licensing server. In fact, Bram discloses multiple

license servers including primary and backup license servers, which are external to the computing cluster.

The system 10 can include *multiple license servers* 940. In some embodiments, each instantiation of the system 10 can include a *primary license server and one or more backup license servers*. For example, if a network executes two versions of the system 10 (e.g., a first version of the system 10 for development and a second version of the system 10 for question and answer processing), *each version of the system 10 can include a separate primary license server and one or more separate backup license servers*. Each license server can be configured to verify that the applications 16 and components communicating with it are included in the version of the system 10 that the license server is associated with.

FEDEX-1006, [0252].

Disaster recovery addresses the problem of keeping the system 10 running through a disaster that can disable a data processing center managing the system 10. The system 10 can address disaster recovery by supporting a geographically distributed configuration that can run in multiple processing sites simultaneously. In some embodiments, the system 10 uses various approaches to provide multi-site processing, such as hot backups, cold backups, and variations in between, which are commonly called warm backups.

FEDEX-1006, [0304].

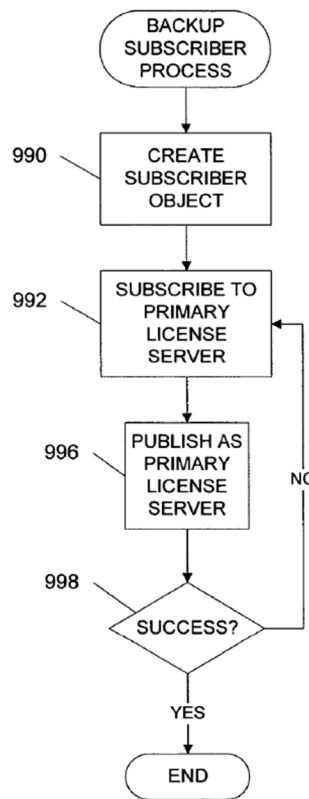
For example, a system user can obtain an updated license control file 950, an updated token 951, and/or an updated license key file 952. In some embodiments, a license control file 950, a token 951, and/or a license key file 952 may be transmitted over a network, such as a local area network (“LAN”) or a wide area network (“WAN”) such as the Internet, to a system user. For example, a license control file 951, a token 951, and/or a license key file 952 can be emailed to a system

user and/or transmitted directly to nodes supporting license servers 940.

FEDEX-1006, [0282]. *See also id.*, [0232] (“Multiple clusters can run simultaneously at multiple locations in order to provide mutual backup options for a single system”), [0257]-[0259], Fig. 28G.

121. It would have been obvious to a POSITA, in my opinion, to modify Sparks’s control node based on Bram’s disclosures on primary and backup licensing servers, such that the modified control node, acting as a backup licensing server, receives licensing information (e.g., floating license information and updated floating license information) from Bram’s primary licensing server. Bram discloses for example:

- “the license server 940 can set itself as a backup server and can subscribe (e.g., create a subscriber process or thread) to the license server that has published itself as the primary license server” (FEDEX-1006, [0257]),
- “the backup license server receives floating license information and updated floating license information from the primary license server” ([0258]),
- If the backup license server successfully publishes itself as the new primary license server (step 998), the backup license server becomes the new primary license server and can exit or terminate the subscriber object it previously created ([0259]),
- Fig. 28G:



(Fig. 28G).

122. A POSITA would have been motivated to modify Sparks’s control node to communicate with one or more external servers for backup, redundancy, and/or disaster recovery reasons. FEDEX-1006, [0232] (“Multiple clusters can run simultaneously at multiple locations in order to provide mutual backup options for a single system”), [0252] (“each version of the system 10 can include a separate primary license server and one or more separate backup license servers”), [0304] (“The system 10 can address disaster recovery by supporting a geographically

distributed configuration that can run in multiple processing sites simultaneously”).

This is in addition to the reasons I discuss in Section IX.C.

C. Claim 3

[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.

123. It is my opinion that the Sparks-Bram combination renders [3] obvious.

124. The Sparks-Bram combination provides a bundle-type parameter that identifies a size of the cluster as I discuss above for [1a-ii] and [1d]. Sparks describes that this parameter identifies at least one of a minimum or maximum node count threshold for the computing cluster. Sparks discloses for example:

- “System administrator 20 may define policies regarding tier characteristics, such as a *minimum number of nodes a tier requires*, ...” (FEDEX-1005, 6:55-67),
- “For each of the tiers, control node 12 *creates a number of “node slots” equal to the maximum number of application nodes 14 that may be deployed*. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot.” (6:22-30),
- “Control node 12 then determines whether *the node count for the tier is met* (74). Control node 12 may, for example, determine *whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities*. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20.” (9:52-62),
- Control node 12 determines whether the *minimum node count for the tier* is met (82).” (10:15-35, *see also* Fig. 4).

D. Claim 4

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

125. It is my opinion that the Sparks-Bram combination renders [4] obvious.

126. Bram discloses activation of applications and/or components if license constraints are satisfied by the current state of the system.

As shown in FIG. 28, the license server 940 also includes a software licensing authentication token (“token”) 951, which can include hardware and/or software. The license server 940 can log on to the token 951 (e.g., access and read data included in the token 951) and can manage access to the token 951. As shown in FIG. 28, the token 951 can include a license key file 952 that defines the licenses available to a particular system user. *The license key file 952 provides authorization to apply the available licenses if constraints of the available licenses are satisfied by the current state of the system 10. Using the license key file 952, the license server 940 can authorize activation of applications 16 and/or components associated with available licenses and can disable (e.g., prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints.* In some embodiments, each license key file 952 can specify parameters and/or constraints for each available license, such as an effective date, an expiration date, a total concurrent activations limit or count, *a list of authorized named nodes, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., “restricted” or “unrestricted”) of components allowed under a license.*

FEDEX-1006, [0245].

Next, *using the basic license included in the floating license information, the license server 940 verifies that the core constraints of the system are satisfied* and that the application 16 is allowed to generate and submit an activation authorization request (step 1024). If

the core constraints are not satisfied (step 1026), the license server 940 can create a reject reply (step 1028) and can send the reject reply to the application 16 (step 1030).

FEDEX-1006, [0273]. *See also* [0255], [0265].

127. Bram discloses various examples of constraints such as those disclosed in, e.g., paragraph [0245] (see above). Bram also discloses that “the license server 940 can check static constraints (e.g., host restrictions) and dynamic constraints (e.g., counts of the number of instances of a component in the system 10 and/or on a specific node) for a required license associated with a component.” *Id.*, [0275]. My discussion above for [1a-ii] which identifies node license parameters in the Sparks-Bram combination is also relevant here.

128. Bram includes several figures (Figs. 28A-28D) showing examples of a license key file which provides node license parameters (e.g., constraints) for multiple licenses (named, e.g., “CORE,” “FRAUD MANAGER,” “ATM HANDLING,” and “BASIC GATEWAY”).

952a

LICENSE KEY FILE	
LICENSE NAME:	CORE
LICENSE MODE:	FULL (UNRESTRICTED)
LICENSE AUTHORIZED HOST(S):	NODE1, NODE2
LICENSE EFFECTIVE DATE:	N/A
LICENSE EXPIRATION DATE:	N/A
EFFECTS:	
	-CORE COMPONENTS ACTIVATED IN UNRESTRICTED MODE
	-CORE COMPONENTS ONLY ALLOWED TO ACTIVATE ON ON NODE1 AND NODE2
	-ATTEMPTS TO ACTIVATE CORE COMPONENTS ON NODES OTHER THAN NODE1 AND NODE2 REJECTED
	-CONFIGURED NON-CORE COMPONENTS ACTIVATED IN UNRESTRICTED MODE
	-AUTHORIZATION COMPONENTS NOT ALLOWED TO ACTIVATE
LICENSE NAME:	FRAUD MANAGER
LICENSE MODE:	EVALUATION (RESTRICTED)
LICENSE AUTHORIZED HOST(S):	NODE1, NODE2
LICENSE EFFECTIVE DATE:	FIRST ACTIVATION
LICENSE EXPIRATION DATE:	60 DAYS FROM FIRST ACTIVATION
LICENSED COPIES:	1
EFFECTS:	
	-FRAUD MANAGEMENT COMPONENTS ACTIVATED IN RESTRICTED MODE
	-FRAUD MANAGEMENT COMPONENTS ONLY ALLOWED TO ACTIVATE ON NODE1 AND NODE2
	-ON FIRST ACTIVATION OF FRAUD MANAGEMENT COMPONENTS EXPIRATION DATE SET IN TOKEN
	-FRAUD MANAGEMENT COMPONENTS VALID FOR 60 DAYS FROM FIRST ACTIVATION
	-ATTEMPTS TO ACTIVATE FRAUD MANAGEMENT COMPONENTS AFTER EXPIRATION DATE REJECTED
	-ATTEMPTS TO ACTIVATE FRAUD MANAGEMENT COMPONENTS ON NODES OTHER THAN NODE1 AND NODE 2 REJECTED
	-ONE FRAUD MANAGEMENT COMPONENT (ONE COPY) ALLOWED TO BE ACTIVATE AT ONE TIME
LICENSE NAME:	ATM HANDLING
LICENSE MODE:	EVALUATION (RESTRICTED)
LICENSE AUTHORIZED HOST(S):	NODE1
LICENSE EFFECTIVE DATE:	N/A
LICENSE EXPIRATION DATE:	N/A
EFFECTS:	
	-ATM HANDLING COMPONENTS ACTIVATED IN RESTRICTED MODE
	-ATM HANDLING COMPONENTS ONLY ALLOWED TO ACTIVATE ON NODE1
	-ATTEMPTS TO ACTIVATE ATM HANDLING COMPONENTS ON NODES OTHER THAN NODE1 REJECTED

FEDEX-1006, Fig. 28C.

952a

LICENSE KEY FILE	
LICENSE NAME: CORE	
LICENSE MODE: FULL (UNRESTRICTED)	
LICENSE AUTHORIZED OS(S): SOLARIS	
LICENSE EFFECTIVE DATE: N/A	
LICENSE EXPIRATION DATE: N/A	
EFFECTS:	
-CORE COMPONENTS ACTIVATED IN UNRESTRICTED MODE	
-CORE COMPONENTS ONLY ALLOWED TO ACTIVATE ON NODES SUPPORTED BY SOLARIS OPERATING SYSTEM	
LICENSE NAME: BASIC GATEWAY	
LICENSE MODE: FULL (UNRESTRICTED)	
LICENSE AUTHORIZED OS(S): SOLARIS	
LICENSE EFFECTIVE DATE: N/A	
LICENSE EXPIRATION DATE: N/A	
LICENSED COPIES: 4	
EFFECTS:	
-BASIC GATEWAY COMPONENTS ACTIVATED IN UNRESTRICTED MODE	
-BASIC GATEWAY COMPONENTS ONLY ALLOWED TO ACTIVATE ON NODES SUPPORTED BY SOLARIS OPERATING SYSTEM	
-UP TO 4 BASIC GATEWAY COMPONENTS (4 COPIES) ALLOWED TO BE ACTIVATE AT ONE TIME	

FEDEX-1006, Fig. 28D. *See also id.*, [0030], [0244], [0245].

129. In my opinion, a POSITA would have understood that Bram's constraints identify one or more bundle-types compatible with the computing node. The "CORE" license provides core functionality, "ATM" provides ATM functionality (as in automated teller machine), the "BASIC GATEWAY" provides gateway functionality. FEDEX-1006, [0052], [0076], Figs. 28C and 28D. As shown in Bram's Fig. 28C, the bundle-type of "CORE" is compatible with nodes 1 and 2. As shown in Bram's Fig. 28D, the bundle-type of "BASIC GATEWAY" is

compatible with any node configured to run the operation system (OS) named SOLARIS.

130. This analysis is further consistent with the '416 Patent which discloses:

The licensing scheme may implement a *multi-tiered (or bundled)* license program. The first part of the license scheme defines a permissible size range for a cluster. In one embodiment, *clusters may be categorized into multiple size ranges, referred to herein as bundle-types*. In alternate embodiments, 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. *A bundle may be sold as a set or group of node or grid types to meet a specific market or function*. For example, a product category may include one or more bundle sizes that served a single purpose such as, e.g., array/LUN purpose. Bundles may have a specific homogeneous purpose along with standard grid parameters such as, e.g., right to use, maximum and minimum node counts, and a specific array type functional brand.

FEDEX-1001, 5:48-64.

As used herein, *compatibility enforcement* may include compatibility between hardware, software, firmware, and/or licensing requirements.

FEDEX-1001, 5:43-45.

E. Claim 5

[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.

131. It is my opinion that the Sparks-Bram combination renders [5] obvious.

132. The Sparks-Bram combination provides for adding one or more available computing nodes to the computing cluster as I discuss above for [1c]. The Sparks-Bram combination further renders obvious assigning a computing node a new node license when the computing node license is incompatible (e.g., license requirement/constraint is not satisfied) with a brand-type associated with the computing cluster (e.g., license requirement/constraint customized to a specific node/cluster).

133. First, Bram discloses how one or more requirements/constraints of a computing node license can be or become unsatisfied. Bram discloses for example:

- “the license server 940 ... *can disable (e.g., prevent activation)* of applications 16 and/or components associated with unavailable licenses or *available licenses that have unsatisfied constraints*” (FEDEX-1006, [0245]),
- “license server [can] *recheck the state of the system 10, the license control file 951, the token 951, and/or the license key file 952*” ([0283]),
- “the primary license server can *determine if any licenses have become effective or expired*” (FEDEX-1006, [0287]), and
- “a component can terminate if a *license required by the component has expired, is no longer available, has been changed, or includes constraints that are no longer satisfied*” ([0293]).
- *See also* FEDEX-1006, [0254], [0278], [0279].

A POSITA would have understood that a computing node license is incompatible as claimed if one or more of the license’s requirements/constraints are not satisfied.

134. Second, in the Sparks-Bram combination, a computing node license assigned to a computing node may be incompatible with its requirements/constraints

if for example the computing node (and its license) was previously assigned to a different tier (computing cluster). My discussion above for [1a-ii], [1b], and [1c] is also applicable here.

135. Sparks discloses that (1) computing nodes can be assigned to different tiers such as a web tier, an application tier, and a database tier and (2) computing nodes may be transferred between tiers, *e.g.*, tier to tier, or tier to free pool to another tier, *etc.*

A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain *e.g.* N-tier application, and ***each tier represents a different computing function.*** A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

FEDEX-1005, 5:48–58.

More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation management (CRM) service. System administrator 20 further defines ***one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain.*** As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a ***web tier, an application tier and a database tier.*** In this manner, distributed computing system 10 may be configured to

automatically *provide web server functions, business application functions and database functions.*

FEDEX-1005, 6:10-21.

After receiving input from system administrator 20 defining the architecture and policy of the enterprise fabric, control node 12 identifies unallocated nodes within free pool 13 that satisfy required node attributes. *Control node 12 automatically assigns unallocated nodes from free pool 13 to respective tier node slots of a tier.* As will be described in detail herein, in one embodiment, control node 12 may assign computing nodes to the tiers in a “best fit” fashion. Particularly, control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20. The assignment of the computing nodes may occur on a tier-by-tier basis beginning with a tier with the highest priority and ending with a tier with the lowest priority.

As will be described in detail below, *control node 12 may automatically add unallocated nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to the free pool when the tier has excess capacity, transfer nodes from tier to tier to meet processing demands, or replace failed nodes with nodes from the free pool.* Thus, computing resources, i.e., computing nodes, may be automatically shared between tiers and domains within the fabric based on user-defined policies to dynamically address high-processing demands, failures and other events.

FEDEX-1005, 7:1-25. *See also id.*, 3:40-48 (“*add an application node to allocated pool 15* to increase processing capacity of distributed computing system 10”), 8:14-25 (“control node 12 may also *add nodes from free pool 13 to a tier when more processing capacity is needed within the tier*, remove nodes from a tier to free pool

13 when a tier has excess capacity, *transfer nodes from tier to tier to meet processing demands*, and replace failed nodes with nodes from the free tier”), 8:61-9:2, 9:28-30.

136. Bram discloses that licenses can be customized to a specific configuration.

The license control file 950, the token 951, and/or the license key file 952 can be encrypted in order to prevent cheating. *Licenses can also be tied to a system user’s specific configuration so that system users cannot share licenses, and distributors cannot serve multiple system users with a single license. By customizing licenses and separating licenses for individual components, licenses can be created that expire on various schedules and have different prices.*

FEDEX-1006, [0247].

137. It would have been obvious to a POSITA to customize licenses to specific nodes/tiers such that they can be priced accordingly, e.g., “have different prices” (FEDEX-1006, [0247]). See also my discussion above in Section IX.C. As an example, a license can be customized to a specific computing node and to a specific tier, e.g., web tier, such that it allows a component to operate on the computing node only if the node is assigned to the web tier. However, the constraints/requirements of that license would not be satisfied or may become unsatisfied, *i.e.*, incompatible, if the node is assigned to a non-web tier (*e.g.*, application or database tier). This node/license customization to a specific tier would

have been recognized by the POSITA as a brand-type associated with the tier (computing cluster).

138. This analysis is further consistent with the '416 Patent which discloses “the node may be ‘branded’ as a licensed node having a specific bundle type.”

If, at operation 520, the node that initiated the license request does not have a node license of its own and there are licenses left in the node license pool, then one of the node licenses may be conferred to the node receiving the management command. Operation 525 may be done in two phases. In the first phase the node license pool may be updated to associate a unique identifier (e.g., a serial number) with the node that is assigned the license. In the second phase, *the node may be “branded” as a licensed node having a specific bundle type*, e.g., by writing a token to a protected non-volatile memory location on the node. *The node license may then be assigned specifically to the node (and may not be assigned to any other node)*. Control may then pass to operation 530, described below.

FEDEX-1001, 7:19-32.

By contrast, if at operation 550 the node is acceptable, then control passes to operation 560 and *it is determined whether the node’s brand is compatible with the cluster as defined by the Right-To-Use characteristics*. In one embodiment, if the node has a *brand* which indicates that the node is no longer part of a different cluster (i.e., if the node has been released from a different cluster) and if its *brand* is valid for inclusion in this cluster’s bundle-type (for example, lower class bundle types cannot join a cluster with a higher class bundle-type, but the reverse is true, e.g., higher class nodes can join lower class clusters), then no additional node license needs to be conferred on the node from the cluster's node license pool. This embodiment allows nodes to be moved between clusters provided both node and cluster bundle-types are compatible.

FEDEX-1001, 7:62-8:8.

139. In my opinion, a POSITA would have recognized that, in the tier reassignment example discussed above, a computing node license assigned to the computing node may become incompatible with a brand-type associated with the computing cluster if reassigned. For example, a new node license may be required, and accordingly assigned, if license requirements of the new tier are different than those of the old tier. As such, it would have been obvious to a POSITA to assign, as needed, a new node license to a computing node that is switching between tiers (computing clusters) to ensure that the computing node continues to provide computing resources when added to the new tier.

F. Claim 6

[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.

140. It is my opinion that the Sparks-Bram combination renders [6] obvious.

141. First, Bram discloses initiating a license check while the computing system is operating as claimed. Bram describes for example:

As described above, in some embodiments, the primary license server can initialize a timer, and once the timer reaches zero or another predetermined count, the timer can alert the license server 940 that it should *recheck the state of the system 10, the license control file 951, the token 951, and/or the license key file 952*. The primary license

server can also initialize additional timers that track time constraints of licenses, which can alert the primary license server to recheck the state of the system 10, the license control file 950, the token 951, and/or the license key file 952 once the timers expire. ***Updating or refreshing the license control file 950, the token 951, and/or the license key file 952 can also cause the primary license server to automatically recheck the state of the system 10, the license control file 950, the token 951, and/or the license key file 952.***

FEDEX-1006, [0283].

In some embodiments, upon receiving a ***recheck request*** from the license server 940, the application licenser component of an application 16 can automatically resubmit an activation authorization request to the license server 940. Once the license server 940 responds to the resubmitted activation authorization request, the application licenser component can update the current license permissions of each component as needed and can instruct each component to recheck its license permissions. ***Each component can recheck its current license permission and can react accordingly. For example, a component can terminate if a license required by the component has expired, is no longer available, has been changed, or includes constraints that are no longer satisfied.***

FEDEX-1006, [0293]. *See also id.*, [0254], [0279], [0287]. As such, a POSITA would have understood that Bram discloses initiating a license check while the computing system is operating.

142. Second, Bram discloses that the license check evaluates at least one of: a license status of the computing node as claimed. Bram describes for example:

- “the primary license server can ***determine if any licenses have become effective or expired***” (FEDEX-1006, [0287]),

- “a component can terminate if a *license required by the component has expired, is no longer available, has been changed, or includes constraints that are no longer satisfied*” ([0293]),
- “the license server 940 can also provide current *license status information* to a system user, a system provider, and/or a system manager for tracking and/or billing purposes” ([0247]), and
- Bram’s disclosure of parameters and/or constraints for each available license:

The license key file 952 provides authorization to apply the available licenses if constraints of the available licenses are satisfied by the current state of the system 10. Using the license key file 952, the license server 940 can authorize activation of applications 16 and/or components associated with available licenses and can disable (e.g., prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints. In some embodiments, *each license key file 952 can specify parameters and/or constraints for each available license, such as an effective date, an expiration date, a total concurrent activations limit or count, a list of authorized named nodes, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., “restricted” or “unrestricted”) of components allowed under a license.*

FEDEX-1006, [0245].

143. Third, Bram discloses that the license check evaluates a node count in the computing cluster as claimed. Bram describes for example:

- “each license key file 952 can specify parameters and/or constraints for each available license, such as ... *a total concurrent activations limit or count, a list of authorized named nodes*” (FEDEX-1006, [0245]),
- “the license server 940 can check static constraints (e.g., host restrictions) and dynamic constraints (e.g., counts of the number of instances of a component in the system 10 and/or on a specific node) for a required license associated with a component” ([0275]).

144. Fourth, the Sparks-Bram combination renders obvious a license check that evaluates a brand-type associated with the computing node. Sparks discloses that computing nodes can be assigned to different tiers such as a web tier, an application tier, and a database tier.

A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

FEDEX-1005, 5:48–58.

More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation management (CRM) service. System administrator 20 further defines *one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain.* As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a *web tier, an application tier and a database tier.* In this manner, distributed computing system 10 may be configured to automatically *provide web server functions, business application functions and database functions.*

FEDEX-1005, 6:10-21.

145. Moreover, Bram discloses that licenses can be customized to a specific configuration.

The license control file 950, the token 951, and/or the license key file 952 can be encrypted in order to prevent cheating. ***Licenses can also be tied to a system user's specific configuration so that system users cannot share licenses, and distributors cannot serve multiple system users with a single license. By customizing licenses and separating licenses for individual components, licenses can be created that expire on various schedules and have different prices.***

FEDEX-1006, [0247].

146. As such, and in addition to the reasons I discuss in Section IX.C, it would have been obvious to a POSITA to implement a license check that evaluates a brand-type associated with the computing node because licenses can be customized to specific tiers/nodes such that they can be priced accordingly. FEDEX-1006, [0247]. For example, a license can be customized (branded) to a specific computing node and to a specific tier, e.g., web tier, such that it allows a component to operate on the computing node only if the node is assigned to the web tier. However, the constraints/requirements of that license would not be satisfied or may become unsatisfied if the node is assigned to a non-web tier (e.g., application or database tier). FEDEX-1006, [0293] (“a component can terminate if a license required by the component has expired, is no longer available, has been changed, or includes constraints that are no longer satisfied”).

147. This analysis is further consistent with the '416 Patent which discloses “the node may be ‘branded’ as a licensed node having a specific bundle type.”

If, at operation 520, the node that initiated the license request does not have a node license of its own and there are licenses left in the node license pool, then one of the node licenses may be conferred to the node receiving the management command. Operation 525 may be done in two phases. In the first phase the node license pool may be updated to associate a unique identifier (e.g., a serial number) with the node that is assigned the license. In the second phase, *the node may be “branded” as a licensed node having a specific bundle type*, e.g., by writing a token to a protected non-volatile memory location on the node. *The node license may then be assigned specifically to the node (and may not be assigned to any other node)*. Control may then pass to operation 530, described below.

FEDEX-1001, 7:19-32.

By contrast, if at operation 550 the node is acceptable, then control passes to operation 560 and *it is determined whether the node’s brand is compatible with the cluster as defined by the Right-To-Use characteristics*. In one embodiment, if the node has a *brand* which indicates that the node is no longer part of a different cluster (i.e., if the node has been released from a different cluster) and if its *brand* is valid for inclusion in this cluster’s bundle-type (for example, lower class bundle types cannot join a cluster with a higher class bundle-type, but the reverse is true, e.g., higher class nodes can join lower class clusters), then no additional node license needs to be conferred on the node from the cluster's node license pool. This embodiment allows nodes to be moved between clusters provided both node and cluster bundle-types are compatible.

FEDEX-1001, 7:62-8:8.

G. Claim 7

[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.

148. It is my opinion that the Sparks-Bram combination renders [7] obvious.

149. As an initial matter, claim 7 recites “the license check comprises ...” however claim 7 is a dependent claim of claim 1 which does not recite “license check.” I note that claim 6 recites a “license check.” Regardless of whether the Patent Owner is allowed to change the dependency of claim 7 or argues that claim 7 requires a license check that comprises the limitations recited therein, claim 7 is invalid over the Sparks-Bram combination for the reasons I discuss herein.

150. Bram discloses that the license server can “recheck the state of the system 10, the license control file 951, the token 951, and/or the license key file 952.”

As described above, in some embodiments, the primary license server can initialize a timer, and once the timer reaches zero or another predetermined count, the timer can alert the license server 940 that it should *recheck the state of the system 10, the license control file 951, the token 951, and/or the license key file 952*. The primary license server can also initialize additional timers that track time constraints of licenses, which can alert the primary license server to recheck the state of the system 10, the license control file 950, the token 951, and/or the license key file 952 once the timers expire. *Updating or refreshing the license control file 950, the token 951, and/or the license key file 952 can also cause the primary license server to*

automatically recheck the state of the system 10, the license control file 950, the token 951, and/or the license key file 952.

FEDEX-1006, [0283].

In some embodiments, upon receiving a *recheck request* from the license server 940, the application licenser component of an application 16 can automatically resubmit an activation authorization request to the license server 940. Once the license server 940 responds to the resubmitted activation authorization request, the application licenser component can update the current license permissions of each component as needed and can instruct each component to recheck its license permissions. *Each component can recheck its current license permission and can react accordingly. For example, a component can terminate if a license required by the component has expired, is no longer available, has been changed, or includes constraints that are no longer satisfied.*

FEDEX-1006, [0293].

Using the license key file 952, the license server 940 can authorize activation of applications 16 and/or components associated with available licenses and *can disable (e.g., prevent activation)* of applications 16 and/or components associated with unavailable licenses or available licenses that have *unsatisfied constraints*.

FEDEX-1006, [0245].

Next, *using the basic license included in the floating license information, the license server 940 verifies that the core constraints of the system are satisfied and that the application 16 is allowed to generate and submit an activation authorization request* (step 1024). *If the core constraints are not satisfied* (step 1026), *the license server 940 can create a reject reply (step 1028) and can send the reject reply to the application 16 (step 1030)*. In some embodiments, the license server 940 can digitally sign the reply, and the application 16 and/or

the application licenser component can use the digital signature to verify that the reply is from an authorized license server. The reject reply can include an “activation declined” response, which can inform the application 16 that authorization is not available for activating any of the suspended components. The reject reply can also inform the application 16 that it should exit or terminate.

FEDEX-1006, [0273]; *see also id.*, [0254], [0279], [0287], FIG. 28J.

151. In my opinion, a POSITA would have understood that Bram discloses, or it would have obvious to perform, a license check comprising implementing an error routine that inhibits operation (*e.g.*, “disable”, “prevent activation,” send a “reject reply”) of the computing node when an incompatibility (*e.g.*, “constraints are not satisfied” / “constraints that are no longer satisfied”) between the computing node and the computing cluster is detected.

H. Claim 8

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

152. It is my opinion that the Sparks-Bram combination renders [8] obvious.

153. Sparks discloses adding an additional computing node to a tier (computing cluster) to meet increased processing demands after the tier has been activated. Sparks describes for example:

FIG. 5 is a flow diagram illustrating exemplary operation of control node 12 when *adding an additional computing node to a tier to meet increased processing demands*. Initially, control node 12 or system

administrator 20 identifies a need for additional processing capacity on one of the tiers (90). Control node 12 may, for example, identify a high processing load on the tier or receive input from a system administrator identifying the need for additional processing capacity on the tier.

Control node 12 then determines whether there are any computing nodes in the free pool of nodes that meet the minimum node requirements of the tier (92). *When there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and assigns the node to the tier (95).*

As described in detail with respect to FIG. 4, control node 12 may determine whether there are any nodes that have node attributes that are an exact match to the node requirements of the tier. If an exact match is found, the corresponding computing node is assigned to a node slot of the tier. If no exact match is found, control node 12 computes the processing energy for each node and assigns the computing node with the minimum processing energy to the tier. Control node 12 remotely powers on the assigned node and remotely boots the node with the image instance associated with the node slot. Additionally, the booted computing node inherits the network address associated with the node slot.

FEDEX-1005, 10:35-62. *See also* Fig. 5.

In one embodiment, a method comprises detecting *a node added to a network* and automatically identifying attributes for the detected node. The method further comprises updating a model to store the attributes for the detected node, wherein the model defines an organization of a distributed computing system having a plurality of computing nodes. The method further comprises automatically *deploying the detected node within the distributed computing system in accordance with the model based on the attributes.*

FEDEX-1005, 1:51-59.

Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated pool 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to *add an application node to allocated pool 15 to increase processing capacity of distributed computing system 10*.

FEDEX-1005, 3:40-48.

I. Claim 9

[9pre] A computing system comprising at least two computing nodes, wherein at least one of the computing nodes comprises:

154. Claim 9 is a system claim that overlaps the method recited in claim 1. The differences in claim language between claims 1 and 9 are addressed below. The limitations of claims 1 and 9 are similar and are therefore rendered obvious by the Sparks-Bram combination for the reasons as I discuss above for claim 1 and for the following reasons.

155. It is my opinion that the Sparks-Bram combination renders [9pre] obvious even to the extent that the preamble is considered a limitation. Sparks discloses a computing system that includes computing nodes e.g., one or more control nodes and application nodes.

FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. *Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide high-*

performance processing.

In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated pool 15 and a maintenance pool 17. In addition, ***distributed computing system 10 includes at least one control node 12.***

Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, ***distributed computing system 10 may include 8 to 512 computing nodes or more.*** Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

FEDEX-1005, 3:7-25.

In general, allocated pool 15 includes application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. ***Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory).*** Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated pool 15.

FEDEX-1005, 3:49-59. *See also id.*, 1:12-21, Fig. 1.

156. Claim 9 recites multiple modules including a cluster module ([9a-i]), node module ([9b-i]), and management module ([9c-i]) --each having a group of functionalities. As I discuss below, a POSITA would have recognized that Sparks

and/or Bram discloses, or at least renders obvious, each of these modules as claimed. Further, Sparks's control node has multiple software modules that provide for respective control node functionalities. FEDEX-1005, 13:52-57 (discussing that the control node includes a "monitoring subsystem 202, a service level automation infrastructure (SLAI) 204, and a business logic tier (BLT) 206"), Fig. 15. Alternatively as well as additionally, it would have been obvious to a POSITA to implement Sparks's control node as having different modules for the claimed functions. It would have been obvious to a POSITA to implement different groups of functionality as separate software modules given the advantages of modular programming in terms of more streamlined development, improved code debugging, and improved testing.

[9a-i] a cluster module stored on a non-transitory computer readable medium, the cluster module configured to manage compatibility operations associated with a computing cluster,

157. It is my opinion that the Sparks-Bram combination renders [9a-i] obvious.

158. Sparks discloses a non-transitory computer readable medium. Moreover, a POSITA would have recognized that Sparks discloses, or renders obvious, a cluster module stored on a non-transitory computer readable medium because instructions corresponding to the functionality described for Sparks's control node can be stored on Sparks's computer-readable media.

Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. ***Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.***

FEDEX-1005, 3:18-25.

In another embodiment, ***a computer-readable medium comprises instructions that cause a processor to*** detect a node added to a network, automatically identify attributes for the detected node, and update a model to store the attributes for the detected node. The model defines an organization of a distributed computing system having a plurality of computing nodes. The instructions further cause the processor to automatically deploy the detected node within the distributed computing system in accordance with the model based on the attributes.

FEDEX-1005, 1:65-2:7. My discussion above for [9pre] regarding software modules is also relevant here.

159. Sparks further discloses that the cluster module is configured to manage compatibility operations (*e.g.*, checking node/tier requirements) associated with a computing cluster (*e.g.*, tier). Sparks describes for example that “[o]ne or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.” FEDEX-1005, 1:33–39. Sparks discloses

that the “[c]ontrol node 12 dynamically assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes.” FEDEX-1005, 9:3-13. A POSITA would have understood that Sparks’s control node checks the compatibility (*e.g.*, checking node/tier requirements) between a node and a tier before assigning the node to the tier. *See also* Section X.A, [1a-i], [1b], [1c].

[9a-ii] the cluster module initializing a computing cluster in the first computing node,

160. It is my opinion that the Sparks-Bram combination renders [9a-ii] obvious. *See* Section X.A, [1b].

[9a-iii] the cluster module adding one or more available computing nodes to the computing cluster;

161. It is my opinion that the Sparks-Bram combination renders [9a-iii] obvious. *See* Section X.A, [1c].

[9b-i] 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,

162. It is my opinion that the Sparks-Bram combination renders [9b-i] obvious.

163. Sparks discloses a non-transitory computer readable medium. Moreover, a POSITA would have recognized that Sparks discloses or at least renders obvious a node module stored on a non-transitory computer readable medium

because instructions corresponding to the relevant functionality of Sparks's control node in the Sparks-Bram combination can be stored on Sparks's computer-readable media.

Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. ***Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.***

FEDEX-1005, 3:18-25.

In another embodiment, ***a computer-readable medium comprises instructions that cause a processor to*** detect a node added to a network, automatically identify attributes for the detected node, and update a model to store the attributes for the detected node. The model defines an organization of a distributed computing system having a plurality of computing nodes. The instructions further cause the processor to automatically deploy the detected node within the distributed computing system in accordance with the model based on the attributes.

FEDEX-1005, 1:65-2:7. *See also* Section X.A, [1a-i], [1a-ii]. My discussion above for [9pre] regarding software modules is also relevant here.

164. Sparks discloses and/or renders obvious that the node module is configured to manage compatibility operations (*e.g.*, storing node/tier requirements/constraints) associated with the at least two computer nodes (*e.g.*, one

or more control nodes and application nodes). *See also* Section §X.I, [9pre]. Sparks's control node "maintain[s] a database having entries for each node slot, where the entries identify the node requirements for each of the tiers" and "retrieves the node requirements for the selected tier from the database." FEDEX-1005, 9:22-27. *See also id.*, 1:51-56, 23:15-20. A POSITA would have understood or it would have been obvious to a POSITA that storing node/tier requirements/constraints are examples of compatibility operations because such requirements/constraints are used to check compatibility. Additional details regarding the stored information, including license information, are provided below in [9b-ii] and [9b-iii].

[9b-ii] the node module storing license information for the computing cluster,

165. It is my opinion that the Sparks-Bram combination renders [9b-ii] obvious. *See* Section X.A, [1a-i].

[9b-iii] 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

166. It is my opinion that the Sparks-Bram combination renders [9b-iii] obvious. *See* Section X.A, [1a-ii].

[9c-i] 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,

167. It is my opinion that the Sparks-Bram combination renders [9c-i] obvious.

168. Sparks discloses a non-transitory computer readable medium. Moreover, a POSITA would have recognized that Sparks discloses or at least renders obvious, a management module stored on a non-transitory computer readable medium because instructions corresponding to the relevant functionality of Sparks's control node in the Sparks-Bram combination can be stored on Sparks's computer-readable media.

Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. ***Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.***

FEDEX-1005, 3:18-25.

In another embodiment, ***a computer-readable medium comprises instructions that cause a processor to*** detect a node added to a network, automatically identify attributes for the detected node, and update a model to store the attributes for the detected node. The model defines an organization of a distributed computing system having a plurality of computing nodes. The instructions further cause the

processor to automatically deploy the detected node within the distributed computing system in accordance with the model based on the attributes.

FEDEX-1005, 1:65-2:7. *See also* Section X.A, [1d]. My discussion above for [9pre] regarding software modules is also relevant here.

169. The Sparks-Bram combination renders obvious that the management module is configured to manage compatibility operations with one or more external servers, as claimed. It would have been obvious to modify Sparks's control node to manage compatibility operations by communicating with one or more external servers to obtain information pertaining to compatibility such as licensing information.

170. Bram discloses multiple servers including primary and backup license servers.

The system 10 can include ***multiple license servers*** 940. In some embodiments, each instantiation of the system 10 can include a ***primary license server and one or more backup license servers***. For example, if a network executes two versions of the system 10 (e.g., a first version of the system 10 for development and a second version of the system 10 for question and answer processing), ***each version of the system 10 can include a separate primary license server and one or more separate backup license servers***. Each license server can be configured to verify that the applications 16 and components communicating with it are included in the version of the system 10 that the license server is associated with.

FEDEX-1006, [0252]. *See also id.*, [0257] (“the license server 940 can set itself as a backup server and can subscribe (e.g., create a subscriber process or thread) to the license server that has published itself as the primary license server”), [0232] (“Multiple clusters can run simultaneously at multiple locations in order to provide mutual backup options for a single system”), [0258], [0259], Fig. 28G.

171. In my opinion, a POSITA would have been motivated to modify Sparks’s control node to communicate with one or more external servers for backup, redundancy, and/or disaster recovery reasons. FEDEX-1006, [0232], [0252], [0304]. This is in addition to the reasons I provided in Section IX.C.

[9c-ii]the management module activating the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.

172. It is my opinion that the Sparks-Bram combination renders [9c-ii] obvious. *See* Section X.A, [1d].

J. Claim 10

[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.

173. It is my opinion that the Sparks-Bram combination renders [10] obvious.

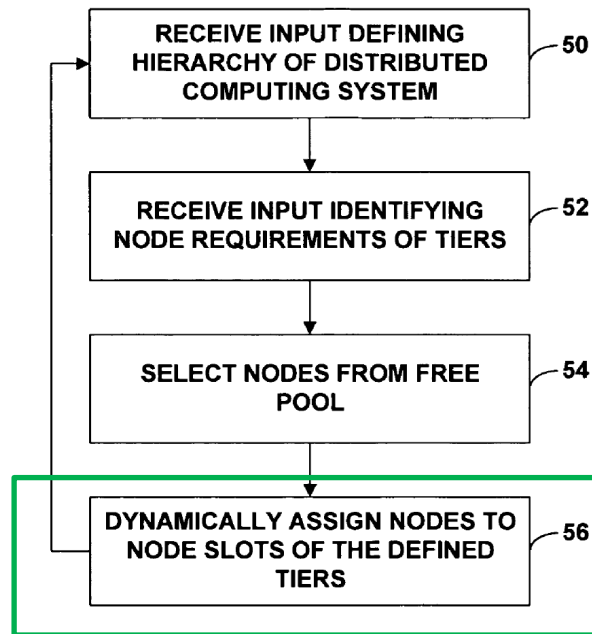
174. Sparks discloses a create cluster module (e.g., control node) to aggregate the at least two computing nodes into a clustered computing system (e.g., tier).

175. The control node of Sparks populates the node slots of a tier (computing cluster), e.g., “*assigns computing nodes to the node slots of each tier* based on the node requirements specified for the tiers and the identified node attributes.”

Control node 12 dynamically *assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes* (56). *Population of the node slots of the tier may be performed on a tier-by-tier basis* beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a “best fit” algorithm.

FEDEX-1005, 9:3-13. This assignment step (56) is part of Fig. 3 which shows the “operation of control node 12 when configuring distributed computing system 10.”

FEDEX-1005, 8:37-39.



FEDEX-1005, Fig. 3 (annotated). *See also* Section X.A, [1c].

176. Sparks describes that the control node aggregates unallocated computing nodes from a free pool into a clustered computing system.

- “Control node 12 automatically *assigns unallocated nodes from free pool 13 to respective tier node slots of a tier.*” FEDEX-1005, 7:1-14.
- “Control node 12 may automatically *add unallocated nodes* from free pool 13 to a tier.” *Id.*, 7:15-25.
- “Additionally, control node 12 may also *add nodes from free pool 13 to a tier when more processing capacity is needed within the tier*, remove nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.” *Id.*, 8:14-25.
- “Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated pool 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to *add an application node to allocated pool 15* to increase processing capacity of

distributed computing system 10.” *Id.*, 3:40-48.

- *See also id.*, 8:61-9:2, 9:28-30.

177. Spark discloses that a tier can be enabled when the node count for that tier is met. FEDEX-1005, 9:14-21 (discussing that “control node 12 selects a tier to enable (60)”), 9:52-62 (“determines whether the node count for the tier is met (74)”), Fig. 4. For example, if the minimum node count is two, then when two computing nodes have been added to a tier, it can be enabled. *See also* Section X.A, [1c], [1d].

K. Claim 11

[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.

178. It is my opinion that the Sparks-Bram combination renders [11] obvious.

179. Bram discloses maintaining a right to use license comprising licensing information for a computing cluster, as claimed. *See also* Section X.A, [1a-i], [1a-ii]. Bram’s describes parameters and/or constraints for each available license in a license key file.

The license key file 952 provides authorization to apply the available licenses if constraints of the available licenses are satisfied by the current state of the system 10. Using the license key file 952, the license server 940 can authorize activation of applications 16 and/or components associated with available licenses and can disable (e.g.,

prevent activation) of applications 16 and/or components associated with unavailable licenses or available licenses that have unsatisfied constraints. In some embodiments, *each license key file 952 can specify parameters and/or constraints for each available license, such as an effective date, an expiration date, a total concurrent activations limit or count, a list of authorized named nodes, a list of authorized operating systems (or operating system aliases), a list of authorized MAC codes, a list of authorized JVMs, and/or an activation mode (e.g., “restricted” or “unrestricted”) of components allowed under a license.*

FEDEX-1006, [0245]. Bram further discloses that “the license server 940 can check static constraints (e.g., host restrictions) and dynamic constraints (e.g., counts of the number of instances of a component in the system 10 and/or on a specific node) for a required license associated with a component.” *Id.*, [0275].

180. Bram further discloses a node license pool comprising licensing information for nodes in the computing cluster. Bram discloses for example, a pool of floating licenses that can be assigned to one or more nodes.

The floating license information can include all licenses and license constraint information currently available to the system 10 (e.g., those licenses purchased and/or obtained by a system user and the related constraints as defined in the license key file 952). In some embodiments, the floating license information also includes current environmental parameters of the system 10, which the license server 940 can use to determine whether constraints of a license are satisfied. For example, the floating license information can include information regarding licenses applied or issued (e.g., permissions granted) to particular applications 16. *The license server 940 can use the floating license information to authorize or decline activation authorization requests generated by components of the system 10 and track applied*

licenses.

FEDEX-1006, [0255]. *See also id.*, [0258], [0264]; Section X.A, [1a-i], [1a-ii].

L. Claim 12

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

181. It is my opinion that the Sparks-Bram combination renders [12] obvious.

182. Sparks discloses an add node module (*e.g.*, control node) to add a computing node to the clustered computing system (*e.g.*, tier). Sparks discloses “*adding an additional computing node to a tier* to meet increased processing demands” and that “[w]hen there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and *assigns the node to the tier* (95).” FEDEX-1005, 10:35-62; Fig. 5. *See also* my discussion above for claims 8, 9, and 10.

M. Claim 13

[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.

183. It is my opinion that the Sparks-Bram combination renders [13] obvious.

184. Bram discloses external licensing servers including primary and backup license servers.

The system 10 can include *multiple license servers* 940. In some embodiments, each instantiation of the system 10 can include a *primary license server and one or more backup license servers*. For example, if a network executes two versions of the system 10 (e.g., a first version of the system 10 for development and a second version of the system 10 for question and answer processing), *each version of the system 10 can include a separate primary license server and one or more separate backup license servers*. Each license server can be configured to verify that the applications 16 and components communicating with it are included in the version of the system 10 that the license server is associated with.

FEDEX-1006, [0252]. *See also id.*, [0257] (“the license server 940 can set itself as a backup server and can subscribe (e.g., create a subscriber process or thread) to the license server that has published itself as the primary license server”), [0232] (“Multiple clusters can run simultaneously at multiple locations in order to provide mutual backup options for a single system”), [0258], [0259], Fig. 28G. *See also* my discussion above for claim 2.

185. Bram discloses that a client, to activate license covered components, broadcasts a get-key request in order to obtain a decryption key from a license server (conditioned on a license check) to decrypt encrypted files (encrypted because they are covered by a license).

Another embodiment provides a system for determining whether a component defined in a computer program is authorized to activate. The system can include a license control file defining a plurality of mappings, each of the plurality of mappings associating at least one of a plurality of components defined in the computer program with at least one required license, and a license server configured to receive an activation authorization request from the component, to determine if the component is associated with at least one required license based on the plurality of mappings, and to provide a response to the component.

FEDEX-1006, [0005]. Bram further discloses:

- “When an application 16 of the system is initialized, the application 16 (e.g., the application licenser component) can broadcast a get-key request” (FEDEX-1006, [0263]),
- “Next, the license server 940 can check that the application 16 that generated the get-key request meets the license constraints of the basic license (step 1006). For example, the license server 940 can verify that the node executing the application 16 that generated the get-key request is a valid node.” ([0265]).
- “After authenticating the get-key request, the license server 940 can obtain a decryption key from the token 951 (step 1008). If the license server 940 successfully obtains the decryption key from the token 951 (step 1010), the license server 940 can format, sign, and send a response, which includes the decryption key, to the application 16 that submitted the get-key request (step 1012).” ([0266]).

186. The request can include a node name/identity and/or enciphered control data.

To complete activation of the suspended components, the application 16 sends an activation authorization request, which includes the license checklist, to the license server 940. In some embodiments, the activation authorization request also includes environmental parameters of the application 16, *such as the name or identifier of a node executing the application 16, which may be needed to determine whether a required license associated with a particular component is available.* The activation authorization request can also include *enciphered control data that the license server 940 can use to authenticate the activation authorization request.* By delaying activation authorization until activation of components of an application 16 is needed, system users can load updated license key files 952 and/or updated license control files 950 in order to add new functionality to the system 10 while the system 10 is running.

FEDEX-1006, [0270].

As shown in FIG. 28J, when the *license server 940 obtains an activation authorization request from an application 16, the license server 940 authenticates the activation authorization request and verifies that the activation authorization request was generated by a valid license server client* (step 1020). In some embodiments, the license server 940 can use the enciphered control data and/or the environmental parameters included in the activation authorization request to authenticate the activation authorization request.

FEDEX-1006, [0271].

187. A POSITA would have recognized that the node name/identity and the enciphered control data are examples of the claimed license key because each can

be used to “unlock” a license, *i.e.*, activate a license covered component. A POSITA would have been motivated to modify Sparks’s control node to communicate with an external licensing server to activate a license covered component for backup, redundancy, and/or disaster recovery reasons. FEDEX-1006, [0232], [0252], [0304]. This is in addition to the reasons I discuss in Section IX.C.

188. Bram’s license server can “check that the application 16 that generated the get-key request meets the license constraints of the basic license” and can “verify that the node executing the application 16 that generated the get-key request is a valid node.” FEDEX-1006, [0265]. After a successful verification, the license server sends a response authorizing the activation of the component --the response includes a decryption key that is used to “decrypt encrypted files (e.g., encrypted JAR files)” associated with the component. *Id.*, *see also* [0243], [0276].

189. In more detail, Bram’s encryption, *e.g.*, Bram’s encrypted JAR files, of a software component prevents it from running/executing. However, the decryption key allows the encrypted JAR files to be decrypted so that they can run/execute. Bram explains that the decryption keys are stored in the software licensing authentication token. FEDEX-1006, [0245]-[0246]. As such, a POSITA would have understood that a decryption key is an example of licensing information.

190. Alternatively as well as additionally, a POSITA would have understand that other information included in Bram's disclosed response are also examples of licensing information:

- Digital signature included in the response. FEDEX-1006, [0273] ("the license server 940 can digitally sign the reply, and the application 16 and/or the application licenser component can use the digital signature to verify that the reply is from an authorized license server"),
- Restriction, included in the response, associated with the activation of a component. FEDEX-1006, [0276] ("[t]he restricted activation authorized response can indicate that a particular component can be activated, but must be activated with limited functionality").

N. Claim 14

[14pre] 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:

191. Claim 14 is an independent computer program product claim that parallels the method claim recited in claim 1. The limitations of claims 1 and 14 are substantially similar and are therefore rendered obvious by the Sparks-Bram combination for the same reasons as I discuss above for claim 1. Any minor differences in claim language between claims 1 and 14 are addressed below.

192. It is my opinion that the Sparks-Bram combination renders [14pre] obvious even to the extent that the preamble is considered a limitation. *See* Section X.A, [1pre]. Sparks 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 operations.

Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. ***Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.***

FEDEX-1005, 3:18-25.

In another embodiment, ***a computer-readable medium comprises instructions that cause a processor to*** detect a node added to a network, automatically identify attributes for the detected node, and update a model to store the attributes for the detected node. The model defines an organization of a distributed computing system having a plurality of computing nodes. The instructions further cause the processor to automatically deploy the detected node within the distributed computing system in accordance with the model based on the attributes.

FEDEX-1005, 1:65-2:7.

In general, allocated pool 15 includes application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. ***Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory).*** Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from

allocated pool 15.

FEDEX-1005, 3:49-59.

[14a-i] store license information for a computing cluster in a memory module associated with a computing node,

193. It is my opinion that the Sparks-Bram combination renders [14a-i] obvious. *See* Section X.A, [1a-i].

[14a-ii] 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;

194. It is my opinion that the Sparks-Bram combination renders [14a-ii] obvious. *See* Section X.A, [1a-ii].

[14b] initialize the computing cluster in a first computing node;

195. It is my opinion that the Sparks-Bram combination renders [14b] obvious. *See* Section X.A, [1b].

[14c] add one or more available computing nodes to the computing cluster; and

196. It is my opinion that the Sparks-Bram combination renders [14c] obvious. *See* Section X.A, [1c]

[14d] activate the computing cluster when the computing cluster includes a number of nodes that complies with the bundle-type parameter.

197. It is my opinion that the Sparks-Bram combination renders [14d] obvious. *See* Section X.A, [1d].

O. Claim 15

[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.

198. It is my opinion that the Sparks-Bram combination renders [15] obvious. *See* Section X.B, [2].

P. Claim 16

[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.

199. It is my opinion that the Sparks-Bram combination renders [16] obvious. *See* Section X.E, [5].

Q. Claim 17

[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.

200. It is my opinion that the Sparks-Bram combination renders [17] obvious. *See* Section X.F, [6].

R. Claim 18

[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.

201. It is my opinion that the Sparks-Bram combination renders [18] obvious. *See* Section X.G, [7].

XI. CONCLUSION

202. I declare that all statements made in this Declaration of my own knowledge are true and that all statements made on information and belief are believed to be true. I further declare that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both (under Section 1001 of Title 18 of the United States Code).

Executed this 14th day of October, 2025.



Dr. Chandrajit Bajaj