

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of: Vipul Pabari Attorney Docket No. 42342-0155IP1
U.S. Patent No.: 10,810,050
Issue Date: October 20, 2020
Appl. Serial No.: 15/201,087
Filing Date: July 1, 2016
Title: VIRTUAL SYSTEMS MANAGEMENT

DECLARATION OF MICHAEL MITZENMACHER

I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable under Section 1001 of Title 18 of the United States Code.

Date: MARCH 10, 2026

By: 
Michael Mitzenmacher

Table of Contents

I.	ASSIGNMENT.....	9
II.	QUALIFICATIONS AND BACKGROUND INFORMATION.....	9
III.	OVERVIEW OF CONCLUSIONS FORMED	15
IV.	LEVEL OF ORDINARY SKILL IN THE ART.....	16
V.	LEGAL STANDARDS	17
	A. Terminology.....	17
	B. Legal Standards.....	18
	C. Anticipation.....	18
	D. Obviousness	19
	E. Enablement	23
	F. Priority Date.....	23
VI.	MATERIALS CONSIDERED	24
VII.	THE ‘050 PATENT	25
	A. Overview of the ‘050 Patent	25
	B. Prosecution History of the ‘050 Patent.....	26
	C. Claim Interpretation.....	37
	D. Priority Date Of Claim 9.....	40
VIII.	OVERVIEW OF THE PRIOR ART	49
	A. Khandekar (MS-1004)	49
	B. Hirschfeld (MS-1006).....	52
	C. Vasilevsky (MS-1005).....	58
	D. Tremain (MS-1009)	60
	E. Pabari (MS-1012)	62
	F. Bonazzola (MS-1013).....	62
IX.	OVERVIEW OF THE PRIOR ART COMBINATIONS	63
	A. The Combination Of Khandekar And Vasilevsky.....	63
	B. The Combination Of Khandekar And Tremain.....	69
	C. The Combination Of Hirschfeld And Vasilevsky	72
	D. The Combination Of Hirschfeld And Tremain.....	77
	E. The Combination Of Pabari And Bonazzola.....	81
X.	THE PRIOR ART ANTICIPATES CERTAIN CLAIMS	86
	A. Ground 1A: Claims 1, 3, 4, 6-8 Are Anticipated By Khandekar	86
	1. Claim 1	86
	2. Claim 3	100
	3. Claim 4	101

4.	Claim 6	103
5.	Claim 7	104
6.	Claim 8	104
XI.	THE PRIOR ART MAKES CERTAIN CLAIMS OBVIOUS	105
A.	Ground 1B: Claims 9, 11, 12, 14, And 16 Are Obvious Over Khandekar 105	
1.	Claim 9	109
2.	Claim 11	139
3.	Claim 12	139
4.	Claim 14	140
5.	Claim 16	141
B.	Ground 1C: Claims 2, 5, 10, And 13 Are Obvious Over The Combination Of Khandekar And Vasilevsky	142
1.	Claim 2	142
2.	Claim 5	143
3.	Claim 10	144
4.	Claim 13	145
C.	Ground 1D: Claims 7 And 15 Are Obvious Over The Combination of Khandekar and Tremain.....	146
1.	Claim 7	146
2.	Claim 15	148
D.	Ground 2A: Claims 1 And 3 Are Obvious Over Hirschfeld	149
1.	Claim 1	149
2.	Claim 3	175
E.	Ground 2B: Claims 2 And 5 Are Obvious Over The Combination of Hirschfeld And Vasilevsky	178
1.	Claim 2	178
2.	Claim 5	180
F.	Ground 2C: Claims 6 And 7 Are Obvious Over The Combination of Hirschfeld And Tremain	182
1.	Claim 6	182
2.	Claim 7	184
G.	Ground 3: Claims 9-16 Are Obvious Over The Combination Of Khandekar And Bonazzola	186
1.	Overview	186
2.	Claim 9	188
3.	Claims 10-16	197
XII.	CONCLUSION.....	199

LISTING OF CLAIMS

Claim 1	
[1.pre]	A method for on demand provisioning of virtual asset via a self-service portal in a virtualization system, the method comprising:
[1.1]	providing a virtualization system that has an inventory that includes physical resources and virtual assets managed by a virtualization layer running on the physical resources;
[1.2]	wherein virtual assets are software instantiations of computer systems;
[1.3]	receiving a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request;
[1.4]	receiving data on inventory available in the virtualization system;
[1.5]	automatically provisioning of a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request;
[1.6]	assigning the provisioned virtual asset to the virtual asset provisioning request; and
[1.7]	providing user access to the provisioned virtual asset.
Claim 2	

[2]	The method of claim 1 further comprising determining one or more performance metrics of the provisioned virtual asset.
Claim 3	
[3]	The method of claim 1 further comprising providing a user interface for submitting the virtual asset provisioning request.
Claim 4	
[4]	The method of claim 3, wherein providing the user interface further comprises providing a user interface to access one or more images in a virtual asset image repository.
Claim 5	
[5]	The method of claim 2, wherein determining the one or more performance metrics further comprises determining one or more real time performance metrics.
Claim 6	
[6]	The method of claim 1, wherein the inventory further comprises one or more application programs.
Claim 7	

[7]	The method of claim 6, wherein one of the application programs comprises one of a network load balancer, a router, a firewall and a storage server.
Claim 8	
[8]	The method of claim 6, wherein one of the application programs comprises a network server.
Claim 9	
[9.pre]	An apparatus that performs automatic management of a virtualization environment in a computer network comprising:
[9.1]	a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device,
[9.2]	the virtualization environment having an inventory that includes network resources and virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets;
[9.3]	the control center computer executing a virtual mapping layer that determines an inventory of network resources and virtual assets available to the virtualization environment,

[9.4]	wherein the virtual mapping layer determines the inventory without regard to processors, operating systems, virtualization platforms, and application software of the virtualization environment; and
[9.5]	the control center computer executing a provisioning manager that communicates with the virtual mapping layer,
[9.6]	the processor being configured to receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request,
[9.7]	automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request,
[9.8]	assign the provisioned virtual asset to the virtual asset provisioning request and provide user access to the provisioned virtual asset.
Claim 10	
[10]	The apparatus of claim 9, wherein the virtualization environment determines one or more performance metrics of the provisioned virtual asset.
Claim 11	

[11]	The apparatus of claim 9, wherein the provisioning manager provides a user interface for submitting the virtual asset provisioning request.
Claim 12	
[12]	The apparatus of claim 11, wherein the provisioning manager provides a user interface to access one or more images in a virtual asset image repository.
Claim 13	
[13]	The apparatus of claim 10, wherein the virtualization environment determines one or more real time performance metrics.
Claim 14	
[14]	The apparatus of claim 9, wherein the inventory further comprises one or more application programs.
Claim 15	
[15]	The apparatus of claim 14, wherein one of the application programs comprises one of a network load balancer, a router, a firewall and a storage server.
Claim 16	
[16]	The apparatus of claim 14, wherein one of the application programs comprises a network server.

I. ASSIGNMENT

1. I have been retained on behalf of Microsoft Corporation (“Microsoft”) to offer technical opinions related to 10,810,050 (“the ’050 patent”) (MS-1001). I understand that Microsoft is requesting that the Patent Trial and Appeal Board (“PTAB” or “Board”) to institute an *inter partes* review (“IPR”) proceeding of the ’050 patent.

2. I have been asked to provide my independent analysis of the ’050 patent based on the prior art publications cited in this declaration.

3. I received no compensation for this declaration beyond my normal hourly compensation based on my time actually spent analyzing the ’050 patent, the prior art publications cited below, and issues related thereto, and I will not receive any added compensation based on the outcome of any IPR or other proceeding involving the ’050 patent.

II. QUALIFICATIONS AND BACKGROUND INFORMATION

4. I am over the age of 18 and am competent to write this declaration. I have personal knowledge, or have developed knowledge of these technologies based upon education, training, or experience, of the matters set forth herein.

5. My qualifications are summarized here and explained in more detail in my curriculum vitae, which is attached as Appendix A to this report. Appendix A also includes a list of my publications and the cases in which I have testified at

deposition, hearing, or trial within the past four years.

6. I am currently employed as a Professor of Computer Science at Harvard University. Specifically, I am the Thomas J. Watson, Sr. Professor of Computer Science in the School of Engineering and Applied Sciences. I joined the faculty at Harvard as an Assistant Professor in January 1999. I was promoted to Associate Professor in 2002 and to Professor in 2005. In 2010, I began a three-year term as Area Dean—equivalent to what other schools call Department Chair—of Computer Science and held that position through June 2013. I also served as Area Co-Chair of Computer Science for the 2018-2019 academic year. From August 1996 to January 1999, I was employed as a Research Scientist at Digital Systems Research Center in Palo Alto, where my work included projects on algorithms for the Internet, including information retrieval on the Web, erasure codes, error-correcting codes, on-line algorithms, and load balancing.

7. I teach courses on various computer science topics to undergraduate and graduate students at Harvard. I have regularly taught the undergraduate class on algorithms and data structures. I have also regularly taught graduate level classes involving algorithms and data structures, that include topics related to machine learning. For example, in both courses I have covered the area of learning-augmented algorithms, where algorithms use predictions to improve their performance.

8. I earned my undergraduate degree in Mathematics and Computer Science from Harvard College in 1991, a Certificate of Advanced Study in Mathematics from Cambridge University in 1992, and a Ph.D. in Computer Science from the University of California at Berkeley in 1996.

9. I have published over 250 research papers in computer science conferences and journals. I am listed as an inventor or co-inventor on 19 issued patents covering computer technology, and I am the co-author of a textbook entitled “Probability and Computing: Randomized Algorithms and Probabilistic Analysis” published by Cambridge University Press. In addition, I have consulted for several companies as a professor, such as Microsoft, Mitsubishi, Akamai, ITA Software, Adverplex, Fluent Mobile, and e-Harmony.

10. I have received numerous professional recognitions from my peers. I am a Fellow of the Association for Computing Machinery (“ACM”) and a Fellow of the Institute of Electrical and Electronics Engineers (“IEEE”), both of which are the two primary professional organizations for computer scientists. ACM Fellows, for example, are described as follows: “ACM’s most prestigious member grade 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.” See <https://awards.acm.org/fellows> (“About ACM Fellows”). My citation for my induction as an ACM Fellow is: “For contributions

to coding theory, hashing algorithms and data structures, and networking algorithms.” Among my awards, I shared in receiving the 2020 ACM Paris Kanellakis Theory and Practice Award for the discovery and analysis of balanced allocations, known as the power of two choices, and their extensive applications to practice.

11. As mentioned, I am the co-author of a textbook entitled “Probability and Computing: Randomized Algorithms and Probabilistic Analysis,” which was published by Cambridge University Press. This textbook is used in advanced undergraduate or beginning graduate courses at Brown University, Harvard University, the University of California at Berkeley, the University of Victoria, Tufts University, the University of Massachusetts at Amherst, Purdue University, the University of Pennsylvania, and several other universities. The second edition added additional sections related to topics in machine learning.

12. I have worked for many years on the foundations and applications of Artificial Intelligence, with recent research examining the ways to improve the efficiency, reliability, and security of modern AI computing systems, including using AI predictions to improve algorithms and data structures. My experience spans large language models (LLMs), distributed and federated learning, security issues, and specific applications such as AI systems and models for biological analyses.

13. My curriculum vitae, included as an appendix to this declaration, includes a list of publications on which I am a named author. It contains further details regarding my experience, education, publications, and other qualifications to render an expert opinion in connection with this proceeding.

14. In writing this Declaration, I have considered the following: my own knowledge and experience, including my work experience in the fields of computer network virtualization, automated deployment of virtual machines, and distributed systems administration; 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:

15. I have reviewed the '050 patent (MS-1001), and excerpts of the prosecution history (MS-1002) of the '050 patent. Among various textbooks, documents, and publications, I have also reviewed the following prior art references:

- U.S. Pat. No. 7,577,722 ("Khandekar") (MS-1004)
- U.S. Pat. App. No. 2005/0044301 ("Vasilevsky") (MS-1005)
- U.S. Pat. No. 6,880,002 ("Hirschfeld") (MS-1006)
- U.S. Pat. App. No. 2002/0049803 ("Bandhole") (MS-1007)
- U.S. Pat. No. 7,457,925 ("Fujino") (MS-1008)

- Int. Pat. Pub. No. WO 02/03220 (“Tremain”) (MS-1009)
- U.S. Pat. App. Pub. No. 2014/0344462 (“Pabari”) (MS-1012)
- U.S. Pat. App. Pub. No. 2015/0347167 (“Bonazzola”) (MS-1013)
- U.S. Provisional App. No. 60/708,473 (MS-1014)
- U.S. Pat. App. No. 15/201,087 (MS-1015)

16. Counsel (Fish & Richardson) has informed me that I should consider these materials through the lens of one of ordinary skill in the art related to the ‘050 patent at the time of the earliest possible priority date of the ‘050 patent, and I have done so during my review of these materials. I have been informed by Counsel to use the date of 8/15/2005 as the Priority Date.

17. I have no financial interest in the outcome of this proceeding. I am being compensated for my work as an expert on an hourly basis. My compensation is not dependent on the outcome of these proceedings or the content of my opinions.

18. My opinions, as explained below, are based on my education, experience, and expertise in the fields relating to the ‘050 Patent. Unless otherwise stated, my testimony below refers to the knowledge of one of ordinary skill in the art as of the earliest possible priority date. Any figures that appear within this document have been prepared with the assistance of Counsel and reflect my understanding of the ‘050 Patent and the prior art discussed below.

III. OVERVIEW OF CONCLUSIONS FORMED

19. This declaration explains the conclusions that I have formed based on my analysis of all of the considered materials and my personal experience. To summarize those conclusions, based upon my knowledge and experience and my review of the prior art references listed above, I believe that:

- Ground 1A: Claims 1, 3, 4, and 6-8 are anticipated by Khandekar
- Ground 1B: Claims 9, 11, 12, 14, and 16 are obvious over Khandekar
- Ground 1C: Claims 2, 5, 10, and 13 are obvious over the combination of Khandekar and Vasilevsky
- Ground 1D: Claims 7 and 15 are obvious over the combination of Khandekar and Tremain
- Ground 2A: Claims 1 and 3 are obvious over Hirschfeld
- Ground 2B: Claims 2 and 5 are obvious over the combination of Hirschfeld and Vasilevsky
- Ground 2C: Claims 6 and 7 are obvious over the combination of Hirschfeld and Tremain
- Ground 3: Claims 9-16 are obvious over the combination of Pabari and Bonazzola

20. In support of these conclusions, I provide an overview of the references in Section X and more detailed comments regarding the anticipation and obviousness of claims 1-16 (“the Challenged Claims”) of the ’050 patent in Section XI.

IV. LEVEL OF ORDINARY SKILL IN THE ART

21. I have been informed that a person of ordinary skill in the art (“POSITA”) is a hypothetical person who is presumed to have the skill and experience of an ordinary worker in the field at the time of the alleged invention. Because I do not know at what date the invention as claimed was made, if ever, I have used the Critical Date of the ’050 patent as the point in time for claim interpretation purposes. My opinion does not change if the invention date is earlier.

22. Based on my knowledge and experience in the field and my review of the ’050 patent and file history, I believe that a person of ordinary skill in the art in this matter would have had (1) a bachelor’s degree in computer science, computer engineering, electrical engineering, or a related field, and (2) two years of industry experience relating to virtual computing systems. A greater amount of education, i.e., a doctorate in an academic area or a related technical field, would compensate for fewer years of work experience.

23. Based on my experiences, I have a good understanding of the capabilities of a POSITA. Indeed, I have taught, participated in organizations, and

worked closely with many such persons over the course of my career. Based on my knowledge, skill, and experience, I have an understanding of the capabilities of a POSITA. For example, from my industry experience, I am familiar with what a POSITA would have known and found predictable in the art. From teaching and supervising my post-graduate students, I also have an understanding of the knowledge that a person with this academic experience possesses. Furthermore, I possess those capabilities myself.

V. LEGAL STANDARDS

A. Terminology

24. I have been informed by Counsel and understand that the best indicator of claim meaning is its usage in the context of the patent specification as understood by one of ordinary skill. I further understand that the words of the claims should be given their plain meaning unless that meaning is inconsistent with the patent specification or the patent's history of examination before the Patent Office. Counsel has also informed me, and I understand that, the words of the claims should be interpreted as they would have been interpreted by one of ordinary skill at the time of the invention was made (not today). The '050 patent was filed July 1, 2016, and claims priority to August 15, 2005, which I refer to as the "Critical Date" in this proceeding. Because I do not know at what date the invention as claimed was made, if ever, I have used the Critical Date of the '050

patent as the point in time for claim interpretation purposes. My opinion does not change if the invention date is earlier.

B. Legal Standards

25. I have been informed by Counsel and understand that documents and materials that qualify as prior art can render a patent claim unpatentable as being anticipated or obvious.

26. I am informed by Counsel and understand that all prior art references are to be looked at from the viewpoint of a person of ordinary skill in the art at the time of the invention, and that this viewpoint prevents one from using his or her own insight or hindsight in deciding whether a claim is anticipated or rendered obvious.

C. Anticipation

27. I understand that patents or printed publications that qualify as prior art can be used to invalidate a patent claim as anticipated or as obvious.

28. I understand that, once the claims of a patent have been properly construed, the second step in determining anticipation of a patent claim requires a comparison of the properly construed claim language to the prior art on a limitation-by-limitation basis.

29. I understand that a prior art reference “anticipates” an asserted claim, and thus renders the claim invalid, if all limitations of the claim are disclosed in

that prior art reference, either explicitly or inherently (i.e., necessarily present).

D. Obviousness

30. I understand that even if a patent is not anticipated, it is still invalid if the differences between the claimed subject matter and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a POSITA.

31. I have been informed by Counsel and understand that a claim is unpatentable for obviousness and that obviousness may be based upon a combination of references. I am informed by Counsel and understand that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. However, I am informed by Counsel and 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.

32. I am informed by Counsel and understand that when a patented invention is a combination of known elements, a court determines whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue by considering the teachings of prior art references, the effects of demands known to people working in the field or present in the marketplace, and the background knowledge possessed by a person having ordinary skill in the

art.

33. I am informed by Counsel and understand that a patent claim composed of several limitations is not proved obvious merely by demonstrating that each of its limitations was independently known in the prior art. I am informed by Counsel and understand that identifying a reason those elements would be combined can be important because inventions in many instances rely upon building blocks long since uncovered, and claimed discoveries almost of necessity will be combinations of what, in some sense, is already known. I am informed by Counsel and understand that it is improper to use hindsight in an obviousness analysis, and that a patent's claims should not be used as a "roadmap."

34. I am informed by Counsel and understand that an obviousness inquiry requires consideration of the following factors: (1) the scope and content of the prior art, (2) the differences between the prior art and the claims, (3) the level of ordinary skill in the art, and (4) any so called "secondary considerations" of non-obviousness, which include: (i) "long felt need" for the claimed invention, (ii) commercial success attributable to the claimed invention, (iii) unexpected results of the claimed invention, and (iv) "copying" of the claimed invention by others.

35. I have been informed by Counsel and understand that an obviousness evaluation can be based on a single reference or a combination of multiple prior art references. I understand that the prior art references themselves may provide a

suggestion, motivation, or reason to combine, but that the nexus linking two or more prior art references is sometimes simple common sense. I have been informed by Counsel and understand that obviousness analysis recognizes that market demand, rather than scientific literature, often drives innovation, and that a motivation to combine references may be supplied by the direction of the marketplace.

36. I have been informed by Counsel and understand that if a technique has been used to improve one device, and a person of ordinary skill at the time of invention would have recognized that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill.

37. I have been informed by Counsel and understand that practical and common sense considerations should guide a proper obviousness analysis, because familiar items may have obvious uses beyond their primary purposes. I have been informed by Counsel and understand that a person of ordinary skill looking to overcome a problem will often be able to fit together the teachings of multiple prior art references. I have been informed by Counsel and understand that obviousness analysis therefore takes into account the inferences and creative steps that a person of ordinary skill would have employed at the time of invention.

38. I have been informed by Counsel and understand that a proper

obviousness analysis focuses on what was known or obvious to a person of ordinary skill at the time of invention, not just the patentee. Accordingly, I understand that any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed.

39. I have been informed by Counsel and understand that a claim can be obvious in light of a single reference, without the need to combine references, if the elements of the claim that are not found explicitly or inherently in the reference can be supplied by the common sense of one of skill in the art.

40. In sum, my understanding is that prior art teachings are properly combined where one of ordinary skill having the understanding and knowledge reflected in the prior art and motivated by the general problem facing the inventor, would have been led to make the combination of elements recited in the claims. Under this analysis, the prior art references themselves, or any need or problem known in the field of endeavor at the time of the invention, can provide a reason for combining the elements of multiple prior art references in the claimed manner.

41. I have been informed by Counsel and understand that in an *inter partes* review, “the petitioner shall have the burden of proving a proposition of unpatentability,” including a proposition of obviousness, “by a preponderance of the evidence.” 35 U.S.C. §316(e).

E. Enablement

42. I have been informed by Counsel and understand that a patent specification must contain a written description of the invention and must enable a person of ordinary skill in the art to make and use the invention. I understand that enablement requires the specification to teach a POSITA how to make and use the full scope of the claimed invention without undue experimentation.

43. I have been informed by Counsel and understand that the enablement requirement applies to the claims as they are written. That is, the specification must enable the scope of what the claims actually recite, not merely a preferred embodiment or a subset of the claimed subject matter.

44. I have been informed by Counsel and understand that when evaluating whether a claim is entitled to the benefit of an earlier filing date in a priority chain, the specification of the earlier-filed application must provide written description and enablement support for the claims as they appear in the later-filed application. If the earlier specification does not provide such support for a particular claim limitation, I understand that the claim is not entitled to the benefit of the earlier filing date with respect to that limitation.

F. Priority Date

45. Counsel has explained to me that in order for a claim of a patent to receive the benefit of a claimed priority date, a person of ordinary skill in the art,

reading the original disclosure associated with the claimed priority date, must immediately discern that the patentee possessed all features described in the claim.

46. Counsel has also explained to me that entitlement to a filing date extends only to the subject matter that is disclosed; not to that which is obvious. The original disclosure must actually or inherently disclose the elements of the claim.

VI. MATERIALS CONSIDERED

47. My analysis and conclusions set forth in this declaration are based on my educational background and experiences in the field (*see* Section II). Based on my knowledge and experience, I am considered to be an expert in the field. Also, based on my knowledge and experience, I understand and know of the capabilities of persons of ordinary skill in the field around the Critical Date.

48. As part of my independent analysis for this declaration, I have considered the following: the background knowledge/technologies that were commonly known to persons of ordinary skill in this art during the time before the Critical Date; my own knowledge and experiences gained from my work experience in the field of the '050 patent and related disciplines; and my experience in working with others involved in this field and related disciplines.

49. In addition, I have analyzed the publications and materials listed above.

50. My analysis and conclusions set forth in this declaration are based on the perspective of a POSITA.

VII. THE '050 PATENT

A. Overview of the '050 Patent

51. The '050 patent describes a “Control Center” that provides “automated management of network-based virtual environments.” MS-1001, 6:38-40. The Control Center computers “manages multiple resources” (e.g., blade servers and other network-based equipment) that “can be configured to support a variety of virtual assets, including desktop computer environments, data storage, application servers, routers, and the like.” *Id.*, 7:2-3, 7:19-24. The Control Center also includes modular application components, including an “Asset Manager” and a “Provisioning Manager,” and also includes a “Virtual Mapping engine” that is “programmed to discover the physical resources” and “also the virtual resources, such as virtual servers, storage, desktops, and the like with which the Control Center has access.” *Id.*, 7:38-52. The Asset Manager provides “visibility and management of all global virtual assets,” including “a global topology view and an inventory of virtual infrastructure from a central console,” and it supports “inventory reports at local, remote, and global context.” *Id.*, 8:4-15.

52. A Provisioning Manager facilitates provisioning through an automated workflow driven by user requests and informed by system inventory.

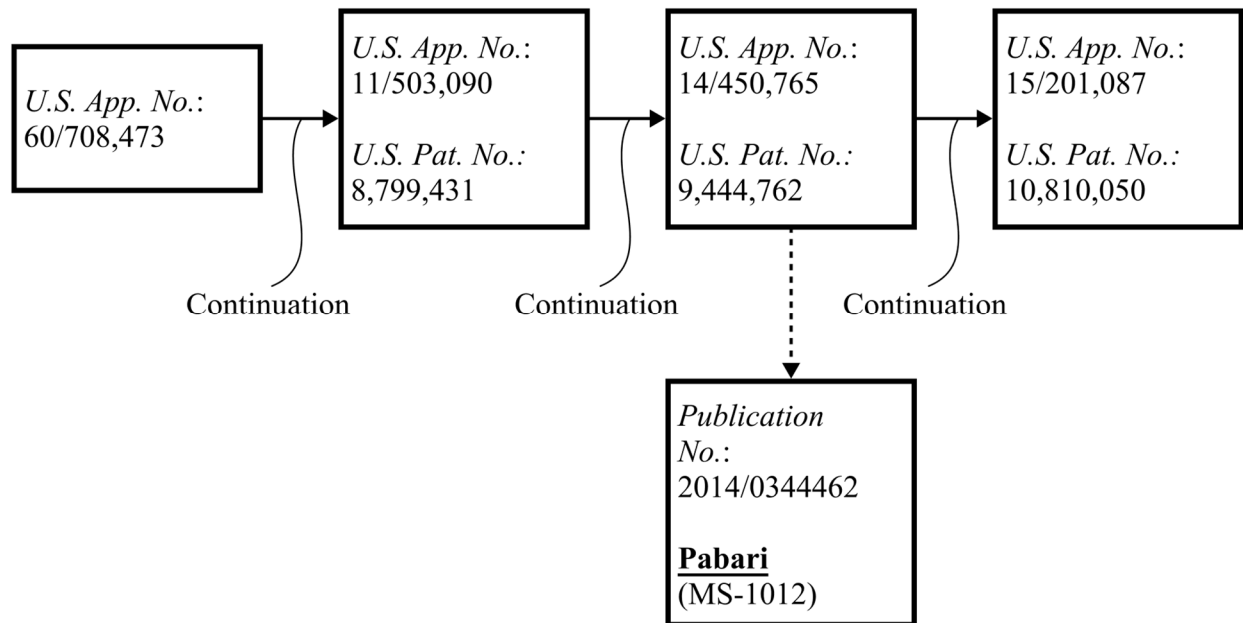
MS-1001, 8:17-26. This includes “an on-demand solution for provisioning of virtual assets for an automated workflow management,” including “allowing users to request and schedule their individual virtual asset needs” and to “prioritize and provision the required assets on demand.” *Id.*, 8:17-22. Provisioning may also be provided in a portal-driven workflow, for instance, through a “portal for users’ virtual asset requests showing available virtual assets,” and enabling approval of provisioning processes while “keeping track of used and available assets.” *Id.*, 8:22-26.

53. The ’050 patent specification’s disclosure is largely focused on an end-to-end, management-plane provisioning workflow within a “Control Center” rather than as a solution uniquely directed to post-provisioning access. MS-1001, 3:45-67. As explained above, the ’050 patent specification includes techniques relating to “a virtual asset provisioning request” and associated sequencing for addressing this request. *Id.*, Abstract, 23:44-45. This sequencing is reinforced by the description of the “Provisioning Manager” as “an on-demand solution for provisioning of virtual assets for an automated workflow management.” *Id.*, 8:17-19.

B. Prosecution History of the ‘050 Patent

54. The ’050 patent issued on October 20, 2020 from U.S. Patent App. No. 15/201,087 (“the ’087 Application”). The ’087 Application was filed on July

1, 2016 as a continuation of U.S. App. No. 14/450,765 (issued as U.S. Pat. No. 9,444,762), which is itself a continuation of U.S. App. No. 11/503,090 (issued as U.S. Pat. No. 8,799,431), and claimed priority to U.S. Provisional Application No. 60/708,473, filed August 15, 2005. MS-1001; Cover. The filing history of the '050 patent is visually shown below:



As shown above, U.S. Pat. No. 9,444,762 (which is a parent patent of the '050 patent) was published as a pre-issuance publication as U.S. Pat. No. 2014/0344462 ("Pabari"; MS-1012).

55. The prosecution of the '087 Application included three Office Actions, all of which presented rejections based U.S. Pat. App. No. 2005/0044301 ("Vasilevsky") and U.S. Pat. App. No. 2003/0236882 ("Yan").

56. The Examiner issued a first Non-final Office Action on April 4, 2017.

MS-1002, 97-109. The Office Action included prior art rejections of claims 1-16 based on Vasilevsky and Yan.

Claim Rejections - 35 USC § 103

5. In the event the determination of the status of the application as subject to AIA 35 U.S.C. 102 and 103 (or as subject to pre-AIA 35 U.S.C. 102 and 103) is incorrect, any correction of the statutory basis for the rejection will not be considered a new ground of rejection if the prior art relied upon, and the rationale supporting the rejection, would be the same under either status.

The following is a quotation of pre-AIA 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claims 1-16 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over **Vasilevsky et al. Pub. No (US. 2005/0044301 A1)** in view of **Yan et al. Pub. No. (US 2003/0236882 A1)**.

Regarding **claim 1**, **Vasilevsky** teaches a method for on demand provisioning of virtual asset via a self-service portal in a virtualization system, the method comprising:

MS-1002, 103

57. The patentee filed a Response to Non-Final Office Action on October 2, 2017. The patentee also amended claim 9 to address rejections cited in the first Non-Final Office Action:

9. (Currently Amended) An apparatus that performs automatic management of a virtualization environment in a computer network comprising:

a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device, the virtualization environment having ~~that has~~ an inventory that includes network resources and ~~[[.]]~~ virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets;

the control center computer executing a virtual mapping layer that determines an inventory of network resources and virtual assets available to the virtualization environment, wherein the virtual mapping layer determines the inventory without regard to processors, operating systems, virtualization platforms, and application software of the virtualization environment; and

the control center computer executing a provisioning manager ~~executed by a processor~~ that communicates with the virtual mapping layer, the processor being configured to receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request, automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request, assign the provisioned virtual asset to the virtual asset provisioning request and provide user access to the provisioned virtual asset.

MS-1002, 131

The Response also included arguments seeking to distinguish claims 1 and 9 from the prior art of record:

Claim 1 recites “providing a virtualization system that has an inventory that includes physical resources and virtual assets managed by a virtualization layer running on the physical resources; wherein virtual assets are software instantiations of computer systems” which is not found in Vasilevsky or Yan.

Yan does not disclose or suggest this claim element and is not relied on in the office action.

Vasilevsky also fails to disclose this claim element. Vasilevsky discloses an operating system layer 103 in a system for providing virtual services. More specifically, the office action explains that the operating system layer 130 may be the claimed virtual assets that are then managed by the virtualization layer 104 in Vasilevsky. The virtualization layer 104 in Vasilevsky isolates applications on a guest operating system (GOS) and maps functions performed by one or more virtual processors to functions performed by one or more physical entities in hardware layer 105. *Vasilevsky at 0058.*

Thus, this claim element is missing from Vasilevsky because: 1) the operating layer 103 of Vasilevsky cannot be the claimed virtual asset that is a software instantiation of computer systems; and 2) the operating layer 103 that is alleged to be the claimed virtual asset is not managed by the virtualization layer identified in Vasilevsky.

Thus, this claim element is not disclosed by Vasilevsky or Yan, alone or in combination.

MS-1002, 133-34

Independent Claim 9

Claim 9 recites “a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device, the virtualization environment having an inventory that includes network resources and virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets” which is not found in Vasilevsky or Yan.

Yan does not disclose or suggest this claim element and is not relied on in the office action.

Vasilevsky also fails to disclose this claim element. Vasilevsky discloses an operating system layer 103 in a system for providing virtual services. More specifically, the office action explains that the operating system layer 130 may be the claimed virtual assets that are then managed by the virtualization layer 104 in Vasilevsky. The virtualization layer 104 in Vasilevsky isolates applications on a guest operating system (GOS) and maps functions performed by one or more virtual processors to functions performed by one or more physical entities in hardware layer 105. *Vasilevsky at 0058*. Furthermore, in Vasilevsky, the operating system layer 103 and the virtualization layer 104 are on the same computer 101 as shown in Figure 1.

Thus, this claim element is missing from Vasilevsky because: 1) the operating layer 103 of Vasilevsky cannot be the claimed virtual asset that is a software instantiation of computer systems; and 2) the operating layer 103 that is alleged to be the claimed virtual asset is not managed by the virtualization layer identified in Vasilevsky.

Thus, this claim element is not disclosed by Vasilevsky or Yan, alone or in combination.

MS-1002, 134

58. The Examiner issued a second Office Action on December 21, 2017.

MS-1002, 14157. This Office Action also included prior art rejections based on Vasilevsky and Yan:

Claim Rejections - 35 USC § 103

6. In the event the determination of the status of the application as subject to AIA 35 U.S.C. 102 and 103 (or as subject to pre-AIA 35 U.S.C. 102 and 103) is incorrect, any correction of the statutory basis for the rejection will not be considered a new ground of rejection if the prior art relied upon, and the rationale supporting the rejection, would be the same under either status.

The following is a quotation of pre-AIA 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claims 1-16 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over **Vasilevsky et al. Pub. No (US. 2005/0044301 A1)** in view of **Yan et al. Pub. No. (US 2003/0236882 A1)**.

MS-1002, 148

59. The patentee filed a Response to Office Action on January 7, 2019. MS-1002, 185-91. While no claims were amended in this response, the patentee submitted similar remarks seeking to distinguish the claims from the prior art of record:

Claim 1 recites “providing a virtualization system that has an inventory that includes physical resources and virtual assets managed by a virtualization layer running on the physical resources; wherein virtual assets are software instantiations of computer systems” which is not found in Vasilevsky or Yan.

Yan does not disclose or suggest this claim element and is not relied on in the office action.

Vasilevsky also fails to disclose this claim element. Vasilevsky discloses an operating system layer 103 in a system for providing virtual services. More specifically, the office action explains that the operating system layer 130 may be the claimed virtual assets that are then managed by the virtualization layer 104 in Vasilevsky. The virtualization layer 104 in Vasilevsky isolates applications on a guest operating system (GOS) and maps functions performed by one or more virtual processors to functions performed by one or more physical entities in hardware layer 105. *Vasilevsky at 0058*.

Thus, this claim element is missing from Vasilevsky because: 1) the operating layer 103 of Vasilevsky cannot be the claimed virtual asset that is a software instantiation of computer systems; and 2) the operating layer 103 that is alleged to be the claimed virtual asset is not managed by the virtualization layer identified in Vasilevsky.

Claim 9 recites “a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device, the virtualization environment having an inventory that includes network resources and virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets” which is not found in Vasilevsky or Yan.

Yan does not disclose or suggest this claim element and is not relied on in the office action.

Vasilevsky also fails to disclose this claim element. Vasilevsky discloses an operating system layer 103 in a system for providing virtual services. More specifically, the office action explains that the operating system layer 130 may be the claimed virtual assets that are then managed by the virtualization layer 104 in Vasilevsky. The virtualization layer 104 in Vasilevsky isolates applications on a guest operating system (GOS) and maps functions performed by one or more virtual processors to functions performed by one or more physical entities in hardware layer 105. *Vasilevsky at 0058*. Furthermore, in Vasilevsky, the operating system layer 103 and the virtualization layer 104 are on the same computer 101 as shown in Figure 1.

Thus, this claim element is missing from Vasilevsky because: 1) the operating layer 103 of Vasilevsky cannot be the claimed virtual asset that is a software instantiation of computer systems; and 2) the operating layer 103 that is alleged to be the claimed virtual asset is not managed by the virtualization layer identified in Vasilevsky.

MS-1002, 189-90

60. Following this response, the Examiner issued a third Office Action on February 26, 2019. MS-1002, 193-208. Claims 1-16 were similarly rejected over Vasilevsky and Yan:

Claims 1-16 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over **Vasilevsky et al. Pub. No (US. 2005/0044301 A1)** in view of **Yan et al. Pub. No. (US 2003/0236882 A1)**.

MS-1002, 200

61. Following the third Office Action, the patentee submitted an Appeal Brief on March 26, 2020. MS-1002, 231-42. The patentee included similar remarks regarding purported distinctions from the prior art of record:

Claim 1 recites “providing a virtualization system that has an inventory that includes physical resources and virtual assets managed by a virtualization layer running on the physical resources; wherein virtual assets are software instantiations of computer systems” which is not found in Vasilevsky or Yan.

Yan does not disclose or suggest this claim element and is not relied on in the office action.

Vasilevsky also fails to disclose this claim element. Vasilevsky discloses an operating system layer 103 in a system for providing virtual services. More specifically, the office action explains that the operating system layer 130 may be the claimed virtual assets that are then managed by the virtualization layer 104 in Vasilevsky. The virtualization layer 104 in Vasilevsky isolates applications on a guest operating system (GOS) and maps functions performed by one or more virtual processors to functions performed by one or more physical entities in hardware layer 105. *Vasilevsky at 0058.*

Thus, this claim element is missing from Vasilevsky because: 1) the operating layer 103 of Vasilevsky cannot be the claimed virtual asset that is a software instantiation of computer systems; and 2) the operating layer 103 that is alleged to be the claimed virtual asset is not managed by the virtualization layer identified in Vasilevsky.

Thus, this claim element is not disclosed by Vasilevsky or Yan, alone or in combination.

MS-1002, 237

Claim 9 recites “a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device, the virtualization environment having an inventory that includes network resources and virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets” which is not found in Vasilevsky or Yan.

Yan does not disclose or suggest this claim element and is not relied on in the office action.

Vasilevsky also fails to disclose this claim element. Vasilevsky discloses an operating system layer 103 in a system for providing virtual services. More specifically, the office action explains that the operating system layer 130 may be the claimed virtual assets that are then managed by the virtualization layer 104 in Vasilevsky. The virtualization layer 104 in Vasilevsky isolates applications on a guest operating system (GOS) and maps functions performed by one or more virtual processors to functions performed by one or more physical entities in hardware layer 105. *Vasilevsky at 0058*. Furthermore, in Vasilevsky, the operating system layer 103 and the virtualization layer 104 are on the same computer 101 as shown in Figure 1.

Thus, this claim element is missing from Vasilevsky because: 1) the operating layer 103 of Vasilevsky cannot be the claimed virtual asset that is a software instantiation of computer systems; and 2) the operating layer 103 that is alleged to be the claimed virtual asset is not managed by the virtualization layer identified in Vasilevsky.

Thus, this claim element is not disclosed by Vasilevsky or Yan, alone or in combination.

MS-1002, 238

62. Following an Examiner Interview on May 29, 2020, the Examiner then issued a Notice of Allowance on June 18, 2020. MS-1002, 263-268. The Examiner identified the following reasons for allowance in the Notice of Allowance:

REASONS FOR ALLOWANCE

2. The following is an examiner's statement of reasons for allowance:

There is no teaching or suggestion in the prior art of the claimed limitations, as presented in the independent claims, specifically providing a virtualization system that has an inventory which includes physical resources and virtual assets managed by a virtualization layer running on the physical resources, receiving a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request and receiving data on inventory available in the virtualization system, automatically provisioning of a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request, and assigning the provisioned virtual asset to the virtual asset provisioning request to provide user access to the provisioned virtual asset as detailed in the dependent claims for such limitations.

MS-1002, 268

C. Claim Interpretation

63. In my opinion, the grounds discussed in Sections X and XI render the Challenged Claims obvious under any reasonable interpretation of the claim language, including plain and ordinary meaning. The analysis set forth in this declaration applies the following constructions discussed with Counsel. The constructions below are consistent with disclosure found in the '050 patent, and demonstrate subject matter that falls within the meaning of these terms. The prior art analysis included below in Sections X and XI are consistent with these constructions.

Claim Term(s)	Proposed Construction
“control center computer”	A computing device that is separate and distinct from the physical host machines being managed and is not itself deployed as a virtual asset <i>See</i> MS-1001, 5:62-6:31, 7:1-14
“automatically provisioning” / “automatically provision”	Determining without human intervention <i>See, e.g.</i> , MS-1001, 4:16-39, 8:21-45
“without regard to...”	Irrespective of any <i>See, e.g.</i> , MS-1001, 4:11-36, 6:42-63
“virtual assets”	Software instantiations of equipment objects that exist only within a virtual environment <i>See, e.g.</i> , MS-1001, 6:32-37
“physical resources”	Physical devices on the network, such as computers, printers, routers, and other tangible hardware <i>See, e.g.</i> , MS-1001, 6:32-37
“apparatus”	A system with two or more devices <i>See, e.g.</i> , MS-1001, 3:50-4:3, 9:45-67
“network resources”	Physical network infrastructure devices, including routers, switches, bridges, infiniband, wireless, radio, optical, and fiber channel equipment

	<i>See</i> MS-1001, 2:10-26
“software instantiations of computer systems”	Software that can virtually perform functions of physical devices that can connect to computers <i>See, e.g.,</i> MS-1001, 3:9-19, 9:62-10:17
“real time performance metrics”	Measurements regarding usage of virtual assets obtained on-demand <i>See, e.g.,</i> MS-1001, 14:12-27
“provisioning” / “provisioned”	Associating a virtual asset to a particular physical computer resource <i>See, e.g.,</i> MS-1001, 20:12-32
“managed” / “manages”	“A manages B” means A does B’s resource planning through one or more of (1) identification and management of network resources and virtual assets, (2) provisioning of virtual assets in response to network workflow demands, (3) dynamic deployment of virtual assets across the computer network, (4) performance measurement and reporting of resources and virtual assets, and (5) planning and forecasting of resource demands and asset utilization of the virtualization environment, such that the functions are carried out without regard to processors, operating systems, virtualization platforms, and application software of the virtualization environment. <i>See, e.g.,</i> MS-1001, 4:16-39

“matches”	Equating to an exact match, not meeting a minimum threshold <i>See, e.g., MS-1001, 18:22-37</i>
-----------	--

D. Priority Date Of Claim 9

64. Based on the understanding specified in Section V.F, claim 9 of the '050 patent is not entitled to a priority date of August 15, 2005 (the filing date of U.S. Provisional Application No. 60/708,473, or MS-1014). The actual priority date of claim 9 is on October 2, 2017, when the patentee amended the original claim 9 that was originally filed with the '087 Application on July 1, 2016, with the amended claim later issuing as claim 9. As discussed in more detail below, this amendment added a “processor” configuration that is now included in claim 9 (in its issued form). This “processor” configuration was not described or suggested in either of MS-1014 (August 15, 2005 Provisional Application) or MS-1015 (July 1, 2016 Utility Application).

65. I have reviewed (1) MS-1014 (August 15, 2005 Provisional Application) and (2) MS-1015 (July 1, 2016 Utility Application). I have also reviewed MS-1012 (November 20, 2014 publication of U.S. Application No. 14/450,765, a prior-filed utility application included in the priority chain of the '050 patent). Based on this review, each document seems to be substantially similar in terms of their written description and their figures. For example, each of

MS-1014, MS-1002, and MS-1012 include the same 18 figures. *See* MS-1012, pp. 2-19; MS-1014, 63-80; MS-1015, 40-57.

66. I have also reviewed MS-1014, MS-1002, and MS-1012 as to how their disclosures specifically relate original claim 9 and amended claim 9 (which later issued as issued claim 9 of the '050 patent).

67. The final issued claim 9 was amended from original claim 9 that was filed with MS-1015 (July 1, 2016 Utility Application). Original claim 9 is reproduced below:

9. An apparatus that performs automatic management of a virtualization environment in a computer network comprising:

a virtualization environment that has an inventory that includes network resources, virtual assets and a virtualization layer that manages the virtual assets;

a virtual mapping layer that determines an inventory of network resources and virtual assets available to the virtualization environment, wherein the virtual mapping layer determines the inventory without regard to processors, operating systems, virtualization platforms, and application software of the virtualization environment; and

a provisioning manager executed by a processor that communicates with the virtual mapping layer, the processor being configured to receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request, automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request, assign the provisioned virtual asset to the virtual asset provisioning request and provide user access to the provisioned virtual asset.

MS-1002, 38

68. As shown above, original claim 9 specified an “apparatus that

performs automatic management of a virtualization environment in a computer network.” The “apparatus” included various elements/limitations, including a “provisioning manager executed by a processor that communicates with the virtual mapping layer.” The same limitation also specifies that “the processor” is configured to (1) “receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request,” (2) “automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request,” (3) “assign the provisioned virtual asset to the virtual asset provisioning request,” and (4) “provide a user access to the provisioned virtual asset.” These four operations are substantially similar to the operations recited in [9.6], [9.7], and [9.8] of issued claim 9.

69. As noted in the image above, original claim 9 described the “processor” and used the transitional phrase “comprising” to include limitations that further included a “processor.” Based on this structure, a POSITA reading original claim 9 would have understood that the “processor” is an element of “apparatus,” and by logical extension, the “processor” is located within the “apparatus.”

70. To understand the scope of “apparatus” recited in original claim 9, a POSITA would have looked to the textual descriptions of MS-1014 (August 15,

2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application).

Based on these documents, a POSITA would have understood that the “apparatus” of claim 9 includes a centralized server that performs provisioning-related operations. This is because MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) describe provisioning and management of virtual environments as being performed “through a Control Center application that provides an interface to virtualization environments in communication with the Control Center computer.” MS-1014, 7:7-20; *see* MS-1014, 6:7-20. The Control Center computer includes components such as a “Provisioning Manager 206” and a “Virtual Mapping engine 216,” both of which correspond to elements of the “apparatus” recited in original claim 9 (i.e., the “provisioning manager” and “virtual mapping layer”).

71. Based on the understanding discussed above, a POSITA would have also understood that the patentee claimed the four operations recited in original claim 9 (“receive...,” “automatically provision...,” “assign...” “provide...” as being performed by a centralized server. This is because MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) each describe a system in which provisioning and management operations are performed by a centralized Control Center computer (e.g., Control Center A 110, Control Center B 112) that “executes application software and thereby manages

one or more virtual environments regardless of any underlying central processing unit (CPU) and regardless of any underlying operating system (OS) or virtualization platform software.” MS-1014, 11:10-12:14; *see* MS-1015, 8:8-9:2.

72. The Control Center includes modular components such as a Provisioning Manager 206, which “provides an on-demand solution for provisioning of virtual assets for an automated workflow management” and provides “a portal for users’ virtual asset requests showing available virtual assets.” MS-1014, 16:15-17:9; *see* MS-1015, 11:19-12:4. The Control Center also includes a Virtual Mapping engine 216 “for determining the physical resources and virtual assets with which the Control Center will communicate.” MS-1014, 14:21-15:15; *see* MS-1015, 10:16-30. These components are described as residing on and executing from the Control Center computer.

73. MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) do not appear to describe any other apparatus that performs these provisioning and management operations. The host machines (computer devices 300) are described as physical resources managed by the Control Center, each hosting a virtualization layer and a Control Agent that “facilitates communication between the external Control Center components 204-214 (FIG. 2) and the Virtualization Layer 322 of the host machine.” MS-1014, 20:4-21; *see* MS-1015, 14:18-15:3. A POSITA would therefore have understood

the “apparatus” of original claim 9 to correspond to the Control Center computer, and the “processor” to be an element of the Control Center computer.

74. As I previously discussed, the features included in issued claim 9 (including those included in [9.1], [9.6], [9.7], and [9.8] were added via amendment in a Response filed on October 2, 2017. The amendment to claim 9 is reproduced below:

9. (Currently Amended) An apparatus that performs automatic management of a virtualization environment in a computer network comprising:

a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device, the virtualization environment having ~~that has~~ an inventory that includes network resources and [[,]] virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets;

the control center computer executing a virtual mapping layer that determines an inventory of network resources and virtual assets available to the virtualization environment, wherein the virtual mapping layer determines the inventory without regard to processors, operating systems, virtualization platforms, and application software of the virtualization environment; and

the control center computer executing a provisioning manager ~~executed by a processor~~ that communicates with the virtual mapping layer, the processor being configured to receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request, automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request, assign the provisioned virtual asset to the virtual asset provisioning request and provide user access to the provisioned virtual asset.

As shown above, the amendment to claim 9 introduced several changes relevant to the “processor” configuration. First, the amendment added the phrase “having a computer device with a processor and memory and a control center computer coupled to the computer device” to what is now [9.1]. This introduced both the “computer device with a processor” and the “control center computer” as distinct elements.

75. The amendment also changed the provisioning manager limitation from “a provisioning manager executed by a processor that communicates with the virtual mapping layer” to “the control center computer executing a provisioning manager...that communicates with the virtual mapping layer.” This change (a) deleted the phrase “executed by a processor” and (b) specified that the control center computer executes the provisioning manager. As a result of these changes, the only remaining antecedent for “the processor” in the provisioning manager limitation is “a processor” introduced in [9.1] as part of the “computer device” (not the “control center computer”).

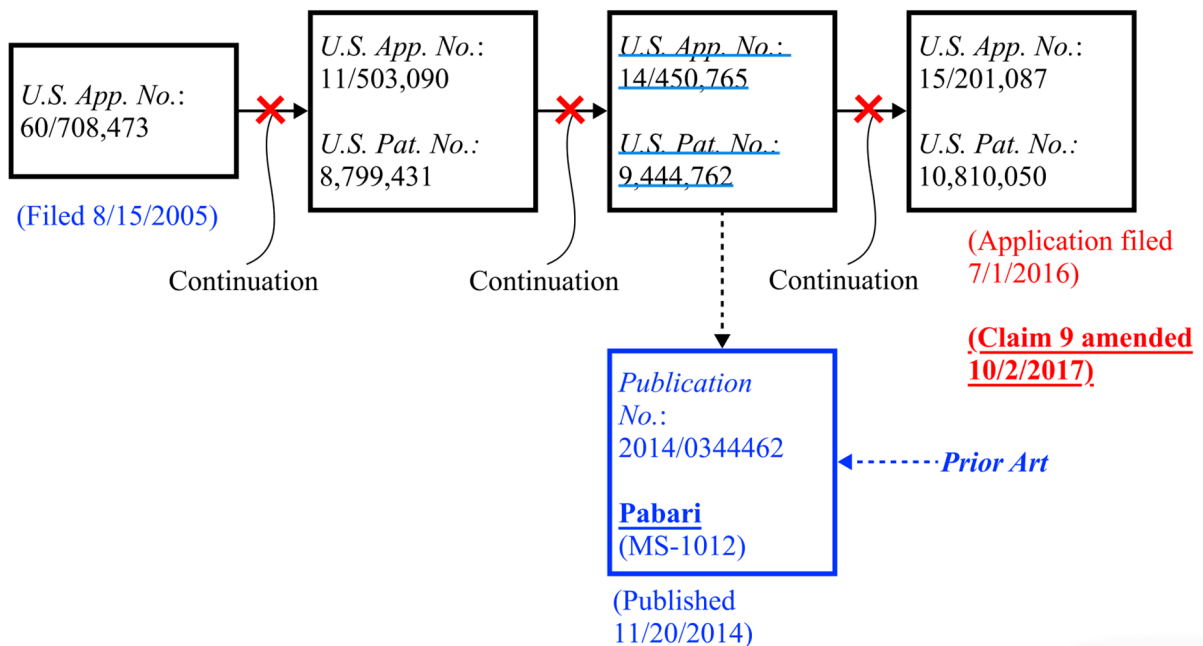
76. Based on my review of MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application), I did not identify any specific description corresponding to the specific amendments to claim 9 discussed above. As I discussed in paragraphs 70-73, MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) attribute all four

provisioning operations to the Control Center computer and its modular components (e.g., Provisioning Manager 206, Asset Manager 204, Virtual Mapping engine 216), and does not describe any other apparatus that performs these operations. Users access the system “through Control Center computers,” which manage virtualization environments “regardless of any underlying central processing unit (CPU) and regardless of any underlying operating system (OS) or virtualization platform software.” MS-1014, 11:10-11:14; MS-1015, 8:8-9:2.

77. Also as I discussed in Paragraph 70-73, the host machines (computer devices 300) are described as physical resources managed by the Control Center, with a Control Agent that performs local tasks at the Control Center's direction (e.g., “the Server makes the decisions and the request is sent to the Control Agent 324 to perform the operation”). MS-1014, 26:4-5; MS-1015, 17:26-27. I did not identify any description within MS-1014 (August 15, 2005 Provisional Application) or MS-1015 (July 1, 2016 Utility Application) of the host machine's processor receiving provisioning requests from users, selecting which virtual asset to provision to which physical resource, tracking the assignment of provisioned assets to requests, or providing user access to provisioned assets. Each of these operations is attributed to the Control Center computer and its components, not to the host machine or its processor. I was also unable to identify a description or suggestion of a configuration in which the host machine's processor is configured

to independently and exclusively perform all four provisioning operations.

78. Based on Counsel's explanation regarding priority entitlement, I understand that this results in claim 9 lacking entitlement to the August 15, 2005 priority date of MS-1014. If there is no entitlement to priority to August 15, 2005, the actual priority date of claim 9 is October 2, 2017 (when the patentee amended claim 9 to include the "processor" configuration similarly present in issued claim 9). As a result, Pabari (MS-1012, which published on November 20, 2014) is available as prior art to claim 9 if its actual priority date is October 2, 2017. This is visually depicted in the chart below:



As I discuss below in Section XI.A.1 (in addressing [9.6-[9.8]]) and Section XI.G.1 (addressing entitlement to priority), earlier applications may describe an architecture from which a POSITA could derive or find obvious a host-side

implementation of provisioning-related functions in a coupled system. But this does not mean those applications demonstrate possession of the later-claimed configuration for priority purposes. For priority entitlement, my understanding is that the question is whether the earlier disclosure itself would have conveyed to a POSITA that the patentee possessed the specific “processor” configuration recited in claim 9. For the reasons discussed above, MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) demonstrate that it did not.

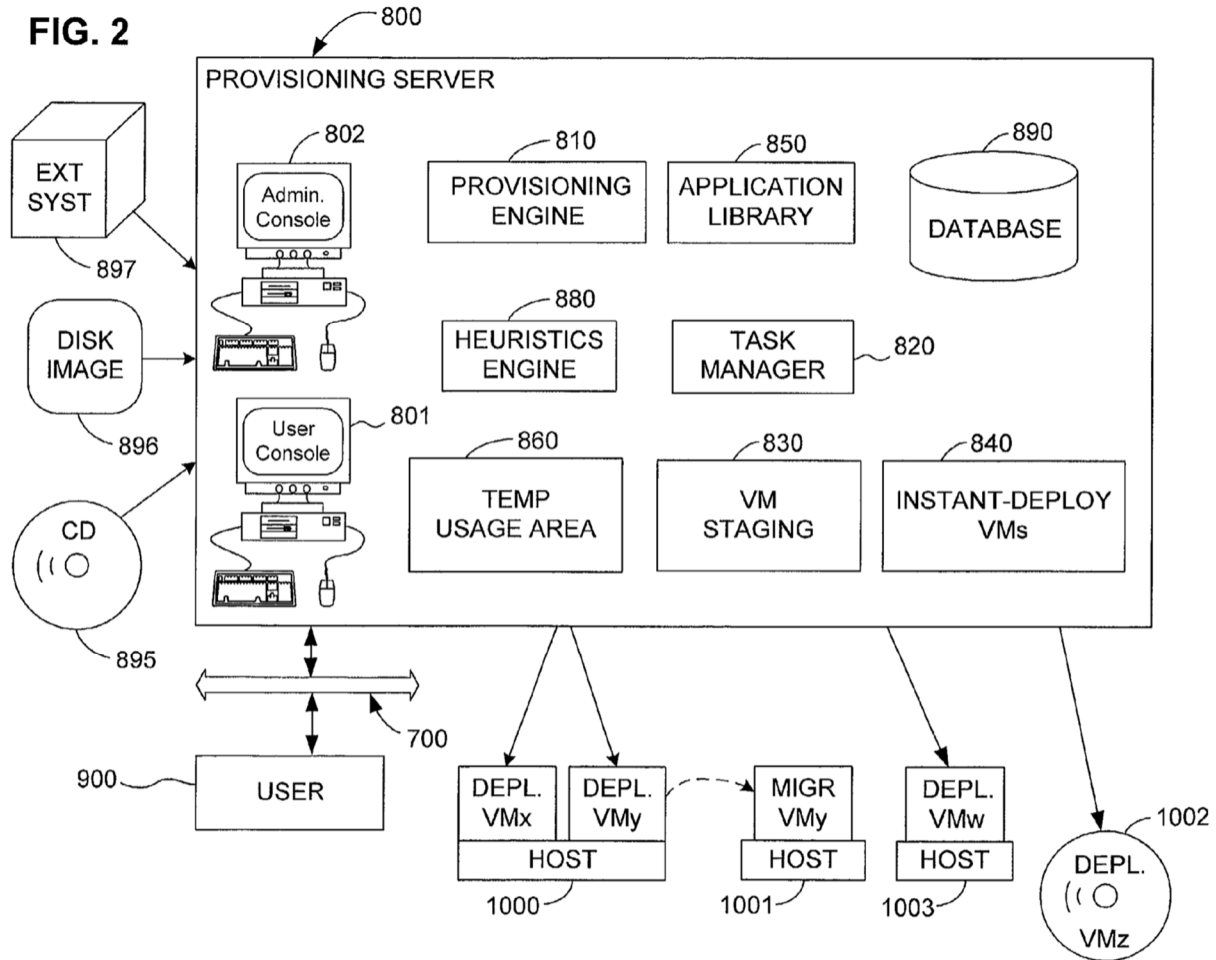
VIII. OVERVIEW OF THE PRIOR ART

A. Khandekar (MS-1004)

79. Khandekar is focused on a VM provisioning architecture in which a user requests a computer having desired configuration characteristics through a centralized provisioning server. MS-1004, 9:65-10:4. Specifically, Khandekar describes that “the user console is preferably a web server,” and the user “interacts with [it] with any conventional web browser,” where the user can “custom-provision a VM” or select a “pre-configured, instantly deployable VM.” MS-1004, 13:66-14:3. Khandekar describes that a user submits configuration selections (e.g., OS configuration such as “hostname, IP address, MAC address”) and selects desired applications, and then “submits the request to the provisioning server,” which applies heuristics/rules to select an appropriate staged VM and an

appropriate host on which to deploy that VM. MS-1004, 9:65-10:42.

80. Figure 2 shows an example of a system 800 used for provisioning:



MS-1004, FIG. 2

81. As shown above, Khandekar's VM provisioning environment allows one or more users 900 to communicate with the provisioning server 800 via a network 700. MS-1004, 9:46-64. The VM provisioning server (e.g., 800) includes a “user console 801” used to “present the various options to the user and to submit the selected options,” and may also include an “administration console 802” for

“monitor[ing] the operational state” and “maintain[ing] the database.” *Id.*, 9:46-64, 10:15-26. The provisioning server receives user input identifying desired configuration characteristics (e.g., operating system selections, application selections, and “OS configuration, for example, hostname, [and] IP address”) and applies a rules/heuristics component (e.g., heuristics engine 880) to “select the most appropriate VM from a library 830 of staged VMs” and to “select[] which host machine 1000, 1001 is most appropriate to provision and deploy the VM on.” *Id.*, 9:65-10:42.

82. Khandekar's host systems provide an execution platform for deployed VMs (e.g., VMx and VMy). Khandekar further describes that “any number of VMs may be deployed and loaded on a single host,” with VMs also capable of being migrated or copied between hosts (e.g., “virtual machine VMy is shown as having been migrated (or copied) from host 1000 to host 1001”). MS-1004, 10:50-60, 12:57-13:8. Workflow components also support provisioning sessions, including a temporary usage area 860 that “holds the user input as well as the various temporary data needed while the provisioning task is in progress.” MS-1004, 10:61-67, 11:1-10. A task manager 820 also initiates a “transaction” and “returns a task identifier” that can be used to “find out the progress of the task.” *Id.*

83. Once the provisioning server provisions a VM on the selected host, Khandekar describes that “the user can then access the newly created VM via any

known remote access software,” including VMware remote console software. MS-1004, 14:4-18. Khandekar also contemplates provisioning requests submitted programmatically, explaining that “software applications acting as a user interact with the provisioning server using a well-defined programmatic interface (API).” *Id.*, 9:46-64.

84. Khandekar is analogous prior art to the '050 patent because it is in the same general field of endeavor, e.g., management of computer network virtualization environments. The '050 patent describes centralized management through a Control Center providing functions, such as building an inventory of physical resources and virtual assets, provisioning virtual assets, and managing the virtualization environment across platforms. Khandekar also addresses centralized VM provisioning in which users interact with a provisioning server via a user console and provision VMs (including “instantly deployable” VMs). *See* MS-1004, 9:65-10:14, 13:66-14:3. Khandekar also describes monitoring deployed VMs and taking management actions, such as migrating VMs based on host load, which reflects similar operational management concerns that would have been important to a POSITA working on the problems addressed in the '050 patent. *See id.*, 13:44-56.

B. Hirschfeld (MS-1006)

85. Hirschfeld describes a “virtualized logical server cloud” in which

users access logical servers through a portal-like user interface without needing to know the access information of the underlying physical servers. MS-1006, 4:37-52, 6:64-7:30, 17:25-39. Although virtualization allows “multiple logical servers to override a physical server's attributes and share its resources” (*id.*, 2:1-7) users historically “had to be aware of and have specific information about the underlying physical server to access the logical servers” (*id.*, 2:8-35), such as the physical server's IP address/server name and port number, and incorrect access inputs could result in access being denied. MS-1006, 4:37-52, 6:64-7:30, 17:25-39.

86. Hirschfeld identifies a desire “to completely separate logical servers from physical servers so that there is no permanent tie between physical and logical resources,” which enables physical servers to function as a resource pool that supports logical servers and allowing a logical server to be reallocated to a different physical server “without users experiencing any change in access approach.” MS-1006, 2:50-59.

87. Hirschfeld also describes a server cloud manager (SCM) that provides a “web page portal or other access mechanism” for remote terminals to communicate with the logical server cloud, such that “a user of any remote terminal” may “login to and access any of the logical servers” via the user interface and SCM. MS-1006, 6:28-63. A user may employ a “local browser to access a web page” associated with the user interface, and the interface provides a “portal or

other interface” through which the user may access one or more logical servers based on permission (e.g., by “point and click” selection of a logical server graphic). *Id.*, 6:64-7:30.

88. Figure 1 shows an overview of a virtualized logical server cloud implemented with a server cloud manager:

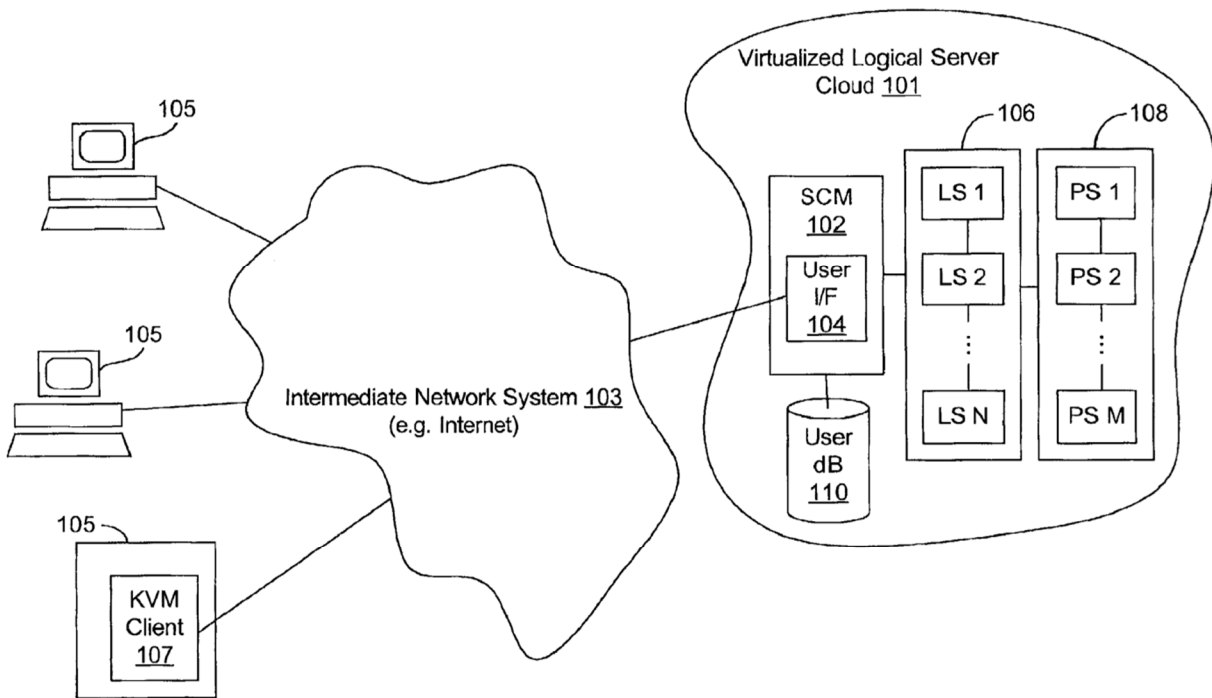


FIG. 1

MS-1006, FIG. 1

89. As shown above, Hirschfeld’s virtualized logical server cloud 101 allows remote terminals 105 to access the system via an intermediate network system 103. MS-1006, 6:28-63. The environment includes one or more “logical servers 106” and a “bank of physical servers 108,” where each logical server “is

linked to and implemented on a corresponding one of the physical servers 108.”

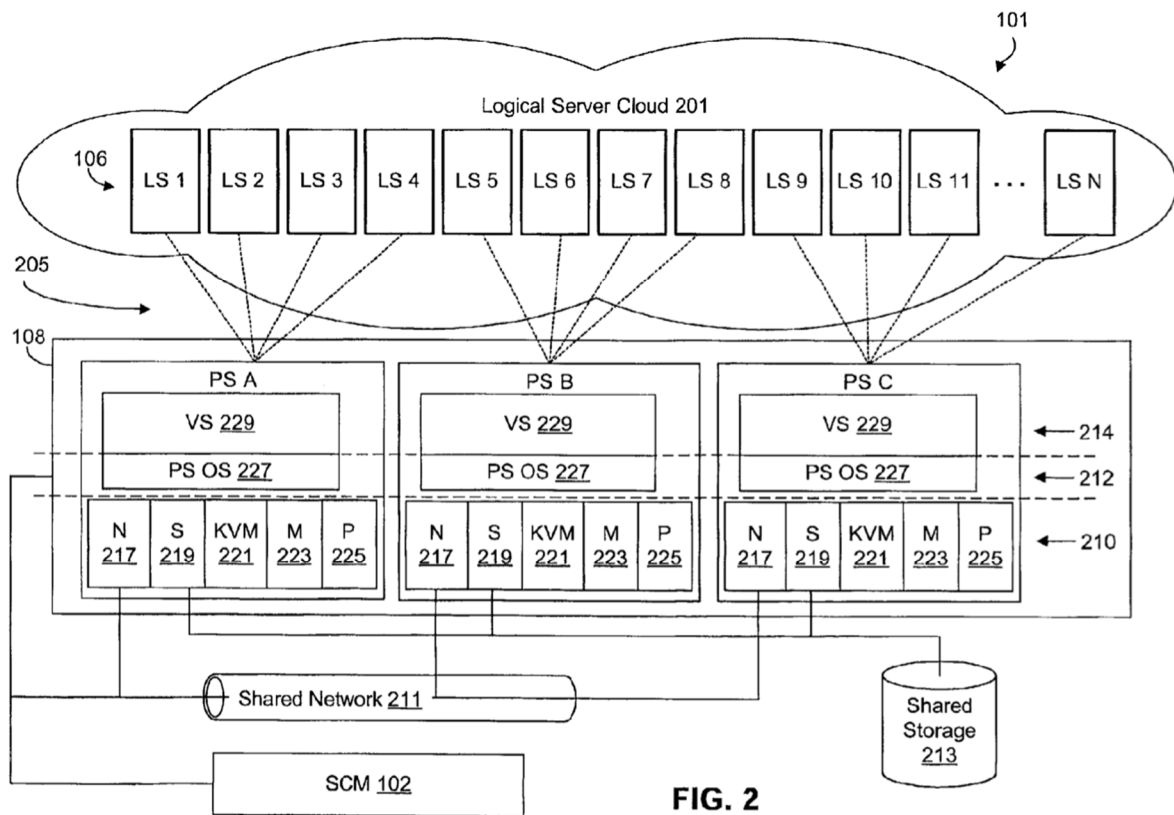
Id., 6:28-63. This architecture includes a server cloud manager (SCM) 102, which “interfaces and controls each of the logical servers 106 and the bank of physical servers 108.” *Id.*, 6:28-63. The SCM 102 includes a “user interface (I/F) 104” that provides a “web page portal or other access mechanism” allowing a user to “login to and access any of the logical servers 106.” *Id.*, 6:28-63.

90. To manage the separation of logical and physical resources, the SCM 102 utilizes a “user database 110” that “incorporates a mapping of users, the logical servers 106 and the physical servers 108.” *Id.*, 7:32-48. This database maintains the “link information that maps each logical server 106 with one of the physical servers 108,” enabling the SCM to automatically access the appropriate physical server without the user needing to supply or be aware of physical server information. *Id.*, 7:32-48; 7:49-62.

91. Hirschfeld also describes a user database that “incorporates a mapping of users, the logical servers and the physical servers,” including “link information that maps each logical server” with a corresponding physical server, such that the SCM “automatically accesses the appropriate physical server corresponding to the selected logical server,” and the user “does not have to supply physical server information” and “does not even need to be aware of” physical-server access/location information. MS-1006, 7:32-62. Hirschfeld emphasizes that its

virtualization “enables a logical server to exist independent of a specific physical server,” with logical attributes “non-deterministically allocated to physical resources,” which creates a cloud of logical servers over one or more physical servers. *Id.*, 17:9-24.

92. Figure 2 illustrates the relationships between the logical servers forming a logical server cloud, the underlying bank of physical servers, and the server cloud manager.



93. EX-1006, FIG. 2

94. As shown above, Hirschfeld depicts a “logical server cloud 201” containing multiple “logical servers 106” that operate above a “bank of physical

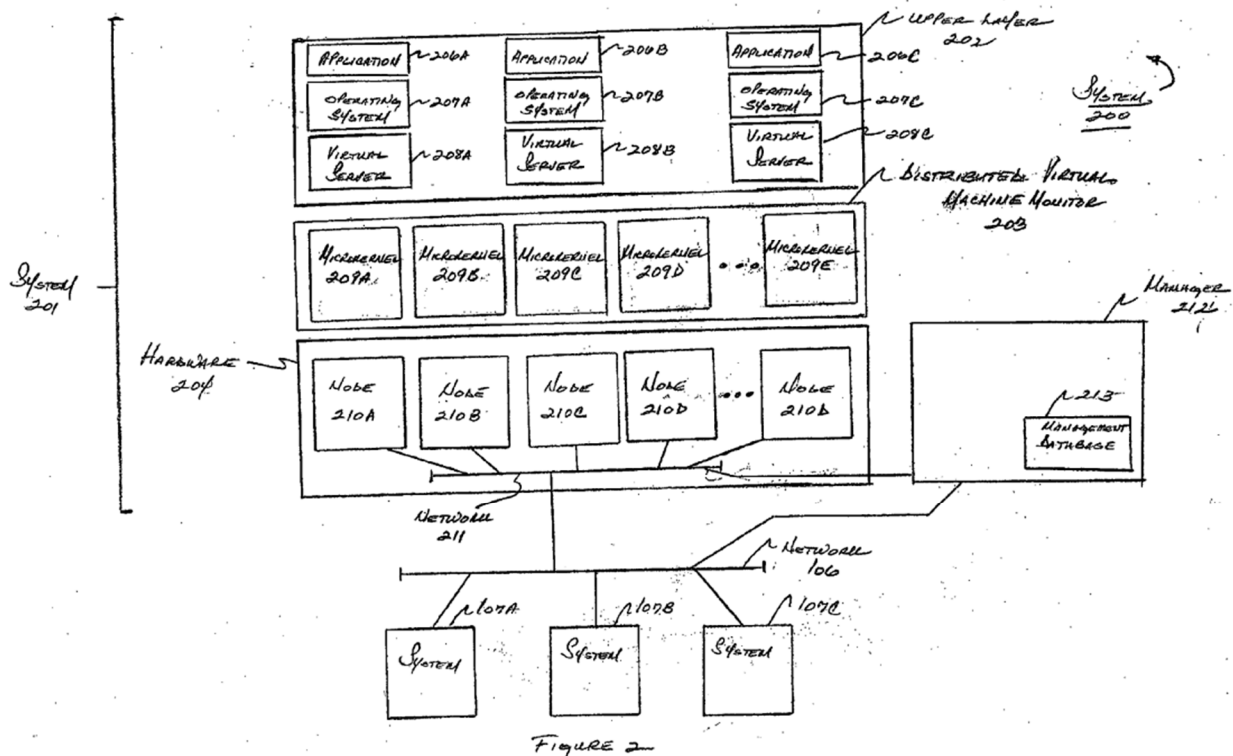
servers 108.” MS-1006, 7:63-8:10. These layers are decoupled, such that “each of the logical servers 106 is non-deterministically linked to a corresponding physical server 108 as illustrated by a corresponding links 205 (illustrated by dashed lines).” *Id.*, 8:1-10. This allows a logical server to be moved to a different physical server “without destroying the identity of the moved logical server 106 as determined by its ‘persistent attributes’.” *Id.*, 8:11-38. The underlying physical servers 108 provide the physical resources, including a “hardware layer 210” (which incorporates a “memory resource ‘M’ 223” and a “processor resource ‘P’ 225”) and an “operating system (OS) layer 212.” *Id.*, 8:39-64. By maintaining the logical servers 106 in a separate cloud 201 connected only by the dashed links 205, Figure 2 visually demonstrates how the system manages logical servers independently from the specific hardware and operating systems of the underlying physical servers. *Id.*, 8:11-38; 8:39-64.

95. Hirschfeld is analogous prior art to the '050 patent because it is in the same general field of endeavor, e.g., virtualization technology and centralized management of virtualized servers. The '050 patent generally relates to management of computer network virtualization environments. Hirschfeld similarly describes a server cloud manager (SCM) for a virtualized logical server cloud. *See* MS-1006, 11:45-57, 11:58-12:19. Hirschfeld also addresses problems aligned with scalable virtualization management (e.g., ensuring user access is

substantially unchanged when a logical server is reallocated to different physical servers) and describes SCM functionality to match demand. See MS-1006, 17:41-49, 18:1-17.

C. Vasilevsky (MS-1005)

96. Vasilevsky describes a virtualization architecture that allows for dynamic resource allocation and management in relation to executing virtual computing systems. MS-1005, [0070], [0072], FIG. 2. Figure 2 shows an example of a system for providing virtual services:



MS-1005, FIG. 2

97. As shown above, system 201 includes a manager 212 that manages the configuration of the system. MS-1005, [0070]. The manager 212 includes an

associated management database 213 that “stores information relating to the configuration of system 201.” *Id.* Manager 212 communicates with “management agent” (not shown) that is “executed by one or more virtual servers” for “the purpose of performing configuration changes, monitoring performance, and performing other administrative functions associated with system 201.” *Id.* This management architecture (including manager 212, management database 213, management agent) enables, for instance, dynamic resource management for virtual servers, including “changing the amount and allocation of resources, both actual and virtual...” *Id.*, [0072].

98. Vasilevsky is analogous prior art to the '050 patent because it is in the same general field of endeavor, e.g., networked computer systems used to provide resources in a data center using virtualization and management infrastructure. The '050 patent is directed to management of computer network virtualization environments. Vasilevsky addresses similar technical problems identified in the '050 patent specification, including that conventional data center configuration changes are often performed manually and can require interruption or reboot. *See* MS-1005, [0004], [0075]-[0079]. Vasilevsky also describes a management architecture employing a manager and management agents (including use of a data store) to control configuration of virtual/distributed servers, which is the same type of virtualization-environment management infrastructure a POSITA would have

consulted when addressing the '050 patent's centralized management objectives. *See id.*, [0071]-[0074], [0080].

D. Tremain (MS-1009)

99. Tremain generally focuses on providing computer services to multiple customers using isolated virtual machines hosted on shared physical computers. MS-1009, Abst. Its disclosure recognizes that hosting multiple organizations on a single real server creates “security, manageability and availability issues,” which creates a need for “enhanced security isolation” to ensure that virtual servers and their data are “held isolated from each other.” *Id.*, 4:6-19, 27:1-26. To address these vulnerabilities, Tremain describes deploying “[e]mbedded ‘virtual security firewalls’” to provide protection to the virtual servers and networks. *Id.*, 32:7-11. These virtual security firewalls also consist of a “special firewall software application” and are “preferably implemented by a virtual machine on the real computer... running firewall software.” *Id.*, 32:13-33:10.

100. Figure 4 illustrates an example of a virtual apparatus arrangement utilizing a virtual security firewall to isolate and protect a customer's virtual network.

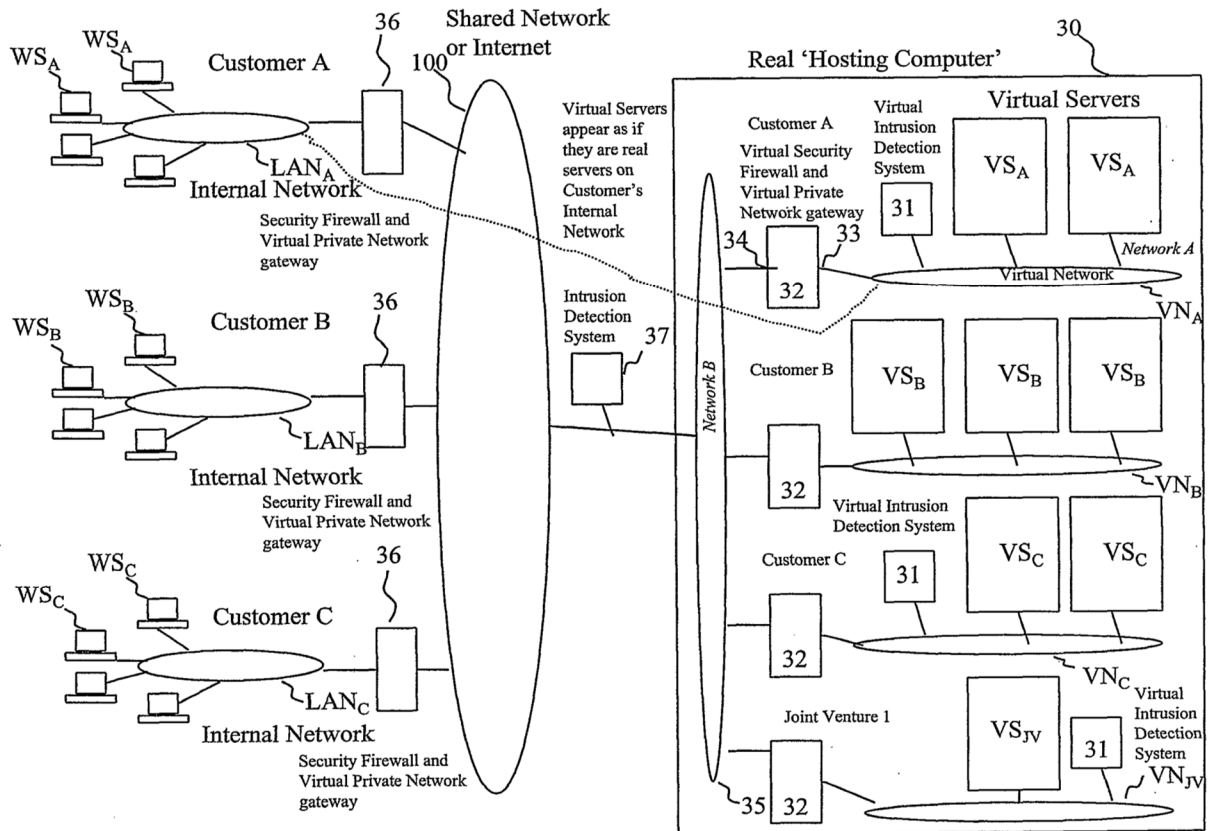


Figure 4

MS-1009, FIG. 4

101. As shown above, Tremain describes creating a virtual server 32 that bridges a customer's virtual network VNA to a virtual "super" network 35. MS-1009, 67:1-26. To secure this environment, "[f]irewall operating systems and software are installed on the virtual server 32 and software is programmed with required access and logging rules." *Id.* This reflects Tremain's technique of deploying a special firewall software application onto a virtual machine to provide enhanced security isolation.

102. Tremain is analogous prior art to the '050 patent because it is in the

same general field of endeavor, e.g., providing and managing virtualized computer services over a network using virtual machines. The '050 patent generally relates to management of computer network virtualization environments. Tremain similarly describes providing computer services to multiple customers by setting up at least one virtual machine for each customer on a real computer. *See* MS-1009, Abst. Tremain also recognizes that conventional approaches to procuring and setting up new equipment and hosting infrastructures involves various challenges, which aligns with the enterprise-scale management challenges that would have been important to a POSITA working on the problems addressed in the '050 patent.

E. Pabari (MS-1012)

103. Pabari is a patent application publication associated with US Patent No. 9,444,762, which is the parent of the '050 patent. I understand that Pabari shares the same disclosure as the '050 patent.

F. Bonazzola (MS-1013)

104. Bonazzola describes a virtualization system in which a setup component 130 executes on a host machine's processor to configure the host machine and locally provision a virtual machine without relying on a separate, dedicated management server. MS-1013, [0011]-[0013], FIGS. 1-3. The setup component 130 performs a configuration process in which a user provides storage,

network, VM, and management system configuration information, and performs an install process in which it uses the resulting installation settings to set up network connectivity, prepare storage, create and start a VM, install an operating system.

Id., [0022]-[0026], [0030]-[0033]. These operations may be performed

“automatically and without user involvement.” *Id.*, [0024].

IX. OVERVIEW OF THE PRIOR ART COMBINATIONS

A. The Combination Of Khandekar And Vasilevsky

105. The Khandekar-Vasilevsky System would have combined Khandekar’s VM provisioning environment with Vasilevsky’s techniques for dynamic resource monitoring and allocation for virtualized resources. In the combined system, Khandekar’s provisioning server environment would be configured to incorporate (or operate alongside) a Vasilevsky-compatible manager and associated management database that communicate with management agents executing on provisioned VMs for performing configuration changes, monitoring performance, and performing other administrative functions. This management architecture would enable the provisioning environment to collect and maintain dynamic performance and loading information for executing virtualized instances (and/or their supporting hosts). The monitored information would be used as part of resource allocation and provisioning decisions in environments where host state information is maintained dynamically and resource reservations are performed via

APIs. *See* MS-1004, 12:57-13:8, 20:54-60; MS-1005, [0070], [0072], [0150].

106. Khandekar provides a provisioning environment in which a provisioning server receives user requests through a console and uses dynamically maintained host state information to select an appropriate host. MS-1004, 12:57-13:8, 20:54-60. Vasilevsky describes a manager (e.g., manager 212) that communicates with management agents “for the purpose of performing configuration changes, monitoring performance, and performing other administrative functions.” MS-1005, [0070]. In the combination, Khandekar’s provisioning server environment would have been configured to incorporate capabilities described Vasilevsky (e.g., for manager 212) so that performance-related state information may be collected and used as part coordinating the provisioning of VM instances. MS-1004, 12:57-13:8, 20:54-60; MS-1005, [0070], [0072].

107. A POSITA implementing Khandekar’s provisioning workflows in larger-scale or more dynamic virtualized environments would have been motivated to look for techniques that (i) make physical-to-virtual resource relationships explicit, (ii) monitor real-time performance metrics, and (iii) permit flexible allocation/reallocation of resources during operation, including through UI/CLI/API-driven management. This is because Khandekar makes clear that its provisioning workflows rely on dynamically-maintained resource state

information. Specifically, “[i]nformation about the current state of the different hosts is preferably input dynamically into a database” used by its heuristics engine and the heuristics engine can “query[] the host via an API” and “[r]equest the host (using the API) to reserve the amount of memory and disk space required.” MS-1004, 12:57-13:8, 20:54-60.

108. A POSITA would have understood that these disclosures (which emphasize dynamic state tracking and API-driven resource reservation during provisioning) create a predictable need for administrable mechanisms for monitoring resource allocation as conditions change (e.g., during operation and across many provisioned assets). This would have led the POSITA to Vasilevsky, which identifies a limitation of conventional approaches and teaches provisioning in real-time without a restart of a virtual computing system. MS-1005, [0076].

109. A POSITA implementing Khandekar’s provisioning workflow would have been motivated to incorporate known virtualization management techniques that provide clearer visibility in how resources are consumed by virtual computing systems. MS-1004, 10:27-42, 21:26-31. Khandekar states provisioning decisions are resource-driven (e.g., a heuristics engine “determine[s] the [host] with enough memory and disk space,” and “request[s] the host (using the API) to reserve the amount of memory and disk space required”), and that provisioning is treated as an orchestrated, transaction-like workflow whose progress is monitored and

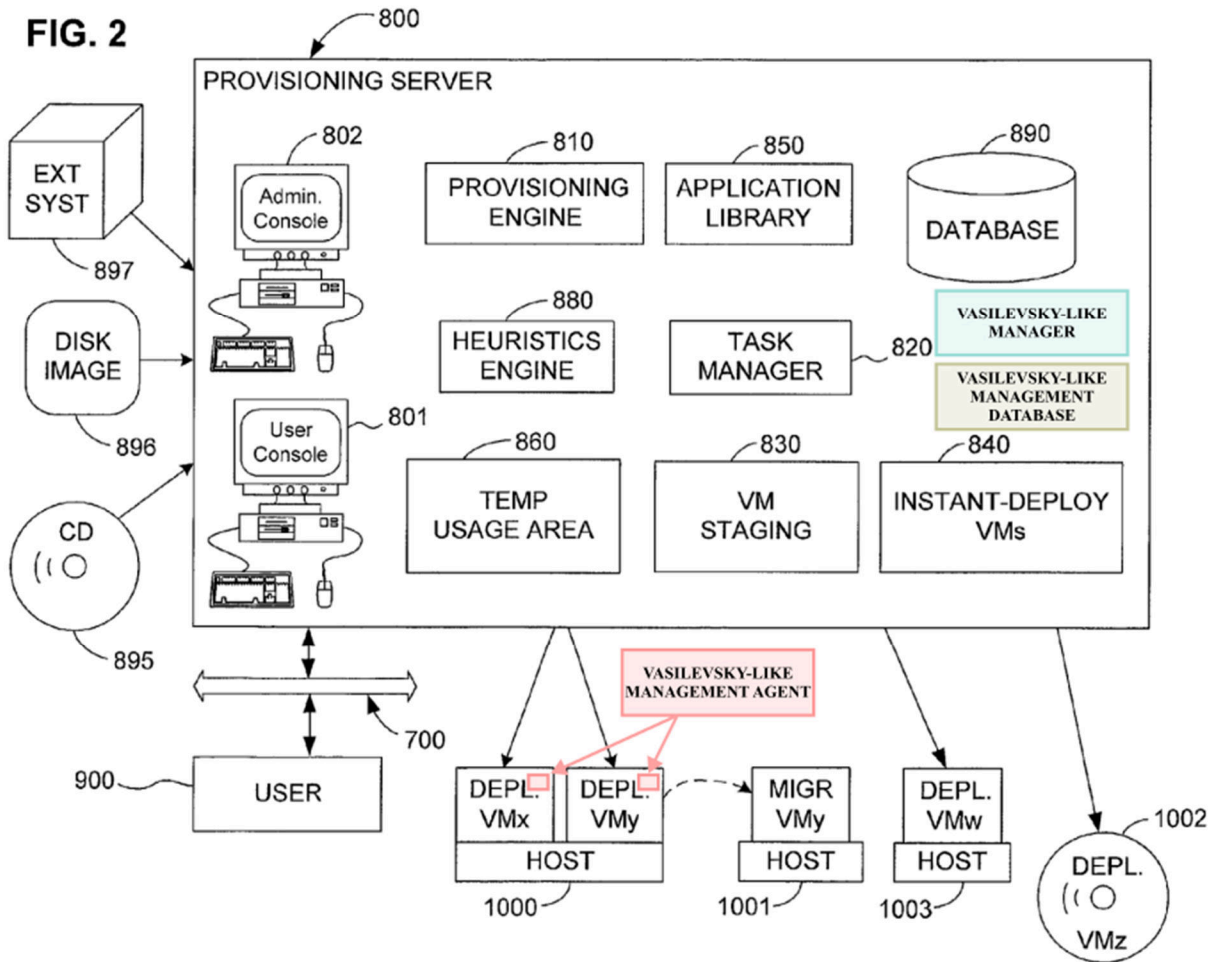
controlled through the provisioning server's management plane. MS-1004, 20:54-21:24.

110. A POSITA would have been motivated to look for established approaches that improve resource monitoring for provisioned VMs. A POSITA would have found Vasilevsky (which describes a management agent that monitors performance metrics and maintains a mapping database) to be pertinent to the Khandekar System. MS-1005, [0084], [0150]. Incorporating those teachings into the system would have been a predictable, compatible refinement to Khandekar's provisioning server (e.g., by incorporating functionality provided by Vasilevsky's manager 212 and management database 213), because it supplies a well-known architecture for monitoring and managing resource allocations (e.g., disk/memory reservation and assignment)—allocations that Khandekar already leverages for its provisioning decisions and provisioning-related tasks. MS-1004, 9:46-64, 20:54-60, 22:53-67. Moreover, to enable monitoring capabilities, each VM provisioned through Khandekar's provisioning environment would similarly incorporate functionality associated with Vasilevsky's management agent, thereby enabling communication with a manager within the provisioning server.

111. In one example implementation, Khandekar's provisioning server (e.g., 800) is configured to include (or operate in conjunction with) a Vasilevsky-compatible manager (e.g., manager 212) and associated management database

(e.g., management database 213). In this example, management database stores and manages configuration and monitoring information for virtualized instances. MS-1004, 9:26-32; MS-1005, [0070]. One or more management agents (executed by provisioned VMs) communicate with the manager “for the purpose of performing configuration changes, [and] monitoring performance,” thereby enabling the collection of performance and loading information associated with provisioned VMs. MS-1005, [0070], [0072].

112. In operation, a user would continue to submit a provisioning request through Khandekar’s user console 801 (e.g., by selecting an “instantly deployable” VM). MS-1004, 9:65-10:14, 13:66-14:3. The provisioning server would select an appropriate host and deploy the requested VM on that host (as described in Khandekar), including using the types of information described in Vasilevsky (e.g., dynamic resource allocation). MS-1004, 10:27-42, 12:57-13:8, 20:54-60; MS-1005, [0072]. After deployment, deployed VMs and hosts “can be monitored,” including monitoring hosts for “load,” and the combined system would use the Vasilevsky-compatible management (and communications with management agents) to receive monitored “performance” and “loading” information for the executing VMs and/or their supporting hosts. MS-1004, 13:44-56; MS-1005, [0070], [0072].



MS-1004, FIG. 2 (modified to incorporate Vasilevsky's teachings)

113. Incorporating Vasilevsky's teachings would have reflected an implementation for representing and controlling how virtualized constructs correspond to underlying physical resources (including during execution), while preserving Khandekar's provisioning-server control plane and provisioning tasks. A POSITA would have also reasonably expected success in doing so because the combination amounts to using known virtualization-management techniques (physical-to-virtual mapping, programmatic allocation) to enhance a known VM

provisioning workflow that already operates through well-defined management interfaces to hosts and already contemplates computer-generated provisioning requests.

B. The Combination Of Khandekar And Tremain

114. As discussed above, Khandekar teaches a provisioning workflow in which a user, through a browser-accessed console, selects (or customizes) a desired VM configuration and submits a provisioning request to a provisioning server. MS-1004, 9:65-10:14, 13:66-14:3. This customization allows the user to select specific software from an “Applications Library 850” to be installed unattended on the newly provisioned VM. *Id.*, 18:57-19:2. Khandekar thereby provides provisioning techniques in which (i) virtual machines act as isolated, “selfcontained” computer systems, and (ii) a control-plane component manages the automated deployment of the user's selected operating systems and application software onto those VMs. *Id.*, 5:25-36, 8:20-35.

115. A POSITA seeking to implement Khandekar's provisioning techniques would have recognized the benefits of enhancing security and network isolation of the deployed VMs, particularly in environments where multiple users or organizations share the underlying physical hosts over a network. Indeed, Khandekar explicitly contemplates these shared multi-user environments since “more than one virtual machine can normally be installed on a single host server.”

MS-1004, 5:37-49. In doing so, Khandekar emphasizes the importance of isolation and access control because shared VMs must be “fully isolated from other virtual machines installed on the same host” to ensure “there will be no loss of security.” *Id.* Khandekar also seeks to control network exposure, noting the ability to dictate whether a provisioned VM “can be reached from any other machine in the LAN” or if it “is not accessible on the LAN, so other machines in the LAN cannot get to it.” *Id.*, 15:1-14. These disclosures would have thereby led a POSITA to seek modifications beyond Khandekar to ensure “no loss of security” and controlling network accessibility for shared hosts.

116. In this respect, Tremain expressly identifies that, in conventional hosting deployments, having “multiple organisations sharing a single applications instance running on a single real server” or “multiple organisations, each with their own applications instance running on a single real server” “has associated security, manageability and availability issues.” MS-1009, 4:6-19. Tremain further describes that it is highly desirable to provide “enhanced security isolation” to ensure that virtual servers and their data are “held isolated from each other.” *Id.*, 27:1-26.

117. In seeking techniques that address these security and isolation issues in a virtualized environment, a POSITA would have found Tremain, which, like Khandekar, also focuses on hosting deployments having “multiple organisations sharing a single applications instance running on a single real server.” MS-1009,

4:6-19. Tremain further describes security features for these configurations by providing “embedded ‘virtual security firewalls’” designed to “provide protection to virtual servers and virtual networks.” *Id.*, 32:7-11. These virtual security firewalls “consist of a special firewall software application which runs on a host operating system on a computer” and are “preferably implemented by a virtual machine on the real computer... running firewall software.” *Id.*, 17:16-19.

118. A POSITA would have viewed Tremain’s software-based virtual firewalls as a compatible enhancement to Khandekar’s application provisioning workflow because both references contemplate deploying software applications onto virtual machines to provide specific functional services to users over a network. *Compare* MS-1004, 9:20-21, 18:57-19:2 (provisioning VMs with customized application software), *and* MS-1009, 17:16-19, 69:22-70:2 (providing computer services by setting up virtual machines running specific software applications, including firewalls).

119. To leverage Tremain’s software-based virtual firewalls within Khandekar’s provisioning framework, a POSITA would have modified the Khandekar System to include the firewall software as an available application program within Khandekar’s Applications Library 850. *See* MS-1004, 11:50-58, 18:57-19:2; MS-1009, 15:16-16:9. This configuration advances Tremain’s stated objective of providing enhanced security isolation and protecting virtual networks

from unauthorized access, while remaining consistent with Khandekar's architecture in which users submit provisioning selections to a centralized server to automatically install and configure desired software applications on their VMs.

MS-1004, 5:25-36, 9:65-10:14, 18:57-19:2; MS-1009, 32:13-33:10, 45:7-23.

120. Finally, a POSITA would have reasonably expected to succeed in combining the teachings of Khandekar and Tremain in the manner discussed above. Both references address virtualized computing environments in which software applications are deployed onto virtual machines hosted on physical servers. MS-1004, 5:25-36, 10:27-42; MS-1009, Abst. The combination simply involves applying Tremain's teaching of a software-based virtual firewall to Khandekar's existing Applications Library workflow, which already contemplates storing and automatically deploying a myriad of software applications (e.g., web servers, databases) onto provisioned VMs. MS-1009, 16:20-17:19, 20:15-21:10, 22:15-23:9; MS-1004, 11:50-58, 18:57-19:2. A POSITA would have understood this as combining known elements (virtual machines, application libraries, and firewall software) according to known methods (automated software installation) to yield predictable results (a securely provisioned VM).

C. The Combination Of Hirschfeld And Vasilevsky

121. The Hirschfeld-Vasilevsky combination would have integrated Hirschfeld's SCM-managed logical server cloud with Vasilevsky's techniques for

dynamic resource monitoring and allocation for virtualized resources. In the combined system, Hirschfeld's SCM framework would be configured to incorporate (or operate alongside) a Vasilevsky-compatible manager and associated management database that communicate with management agents executing on provisioned logical servers for performing configuration changes, monitoring performance, and performing other administrative functions. This management architecture would have enabled the SCM to collect and maintain dynamic performance and loading information for executing logical server instances (and/or their supporting physical servers). The monitored information would be used as part of resource allocation and redistribution decisions in environments where the SCM already tracks resource allocation data and redistributes resources based on demand and usage. *See* MS-1006, 2:63-3:11, 12:21-43; MS-1005, [0070], [0072], [0150].

122. Hirschfeld provides a virtualization environment in which the SCM maintains status/instance information for logical servers, tracks resource allocation data (e.g., “the percentage of CPU time allocated to a particular logical server” and “the amount of memory available to a logical server”), and “may redistribute processor resources based on demand and usage.” MS-1006, 2:63-3:11, 12:21-43.

123. Vasilevsky describes a manager (e.g., manager 212) that communicates with management agents “for the purpose of performing

configuration changes, [and] monitoring performance,” where resources are evaluated and adjusted based on the changing requirements of applications or operating systems, including “performance, loading, etc.” MS-1005, [0070], [0072]. In the combination, Hirschfeld's SCM would be configured to incorporate capabilities described in Vasilevsky (e.g., for manager 212) so that performance-related state information may be collected and used as part of coordinating the management and provisioning of logical server instances. MS-1006, 2:63-3:11, 12:21-43; MS-1005, [0070], [0072].

124. A POSITA implementing Hirschfeld’s SCM-managed logical server cloud would have been motivated to incorporate known virtualization management techniques that provide clearer visibility into how resources are consumed by logical servers executing on physical servers. Hirschfeld’s SCM already tracks resource allocation information (e.g., CPU cycles, memory) and redistributes resources based on demand and usage, but does not specifically describe dedicated mechanisms for collecting and evaluating runtime performance data from executing logical servers. MS-1006, 12:21-43. These disclosures (which emphasize resource tracking and demand-based redistribution) create a predictable need for administrable mechanisms for monitoring resource performance as conditions change (e.g., during operation and across many provisioned logical servers). This would have led a POSITA to Vasilevsky, which identifies a limitation of

conventional approaches and teaches provisioning of computing resources in real-time without a restart of a virtual computing system. MS-1005, [0076].

125. A POSITA would have found Vasilevsky (which describes a management agent that monitors performance metrics and maintains a mapping database) to be pertinent to the Hirschfeld System. MS-1005, [0084], [0150]. Incorporating those teachings into the system would have been a predictable, compatible refinement to Hirschfeld's SCM framework (e.g., by incorporating functionality provided by Vasilevsky's manager 212 and management database 213), because it supplies a well-known architecture for monitoring and managing resource allocations (e.g., allocations that Hirschfeld's SCM already leverages for its resource tracking and redistribution functions). MS-1006, 12:21-43; MS-1005, [0070], [0072]. Hirschfeld's existing agent-based architecture (in which PSAs and LSAs already communicate with the SCM to manage physical and logical attributes of logical servers) also provides a natural integration point for Vasilevsky's management agents, because each logical server provisioned through Hirschfeld's SCM framework would similarly incorporate functionality associated with Vasilevsky's management agent, thereby enabling communication with a manager within the SCM. MS-1006, 3:50-65, 11:4-25, 11:27-43, 11:45-57.

126. In one example implementation, Hirschfeld's SCM is configured to include (or operate in conjunction with) a Vasilevsky-compatible manager (e.g.,

manager 212) and associated management database (e.g., management database 213). In this example, the management database stores and manages configuration and monitoring information for logical server instances. MS-1006, 2:63-3:11; MS-1005, [0070]. One or more management agents (executed by provisioned logical servers) communicate with the manager “for the purpose of performing configuration changes, [and] monitoring performance,” which enables the collection of performance and loading information associated with provisioned logical servers. MS-1005, [0070], [0072].

127. In operation, a user would continue to interact with the SCM through a web page portal to initiate instancing functions (e.g., copy, move) for logical servers. MS-1006, 6:28-63, 7:22-30. The SCM would select a compatible physical server and associate the logical server with that host (as described in Hirschfeld), including using the types of information described in Vasilevsky (e.g., dynamic resource allocation). MS-1006, 8:11-38; MS-1005, [0072]. After provisioning, the SCM would use the Vasilevsky-compatible management architecture (and communications with management agents) to receive monitored “performance” and “loading” information for the executing logical servers and/or their supporting physical servers. MS-1006, 12:21-43; MS-1005, [0070], [0072].

128. Incorporating Vasilevsky's teachings would have reflected an implementation for representing and controlling how virtualized constructs

correspond to underlying physical resources (including during execution), while preserving Hirschfeld's SCM control plane and management functions. A POSITA would have also reasonably expected success in doing so because the combination amounts to using known virtualization-management techniques (performance monitoring, dynamic resource allocation) to enhance a known SCM management framework that already operates through well-defined agent-based interfaces to physical and logical servers and already contemplates resource tracking and redistribution.

D. The Combination Of Hirschfeld And Tremain

129. The Hirschfeld-Tremain combination would have integrated Hirschfeld's SCM-managed logical server cloud with Tremain's teaching of a software-based firewall implemented within a virtual machine. In the combined system, Tremain's virtual firewall software would be made available as an application program within Hirschfeld's logical server cloud, so that logical servers may be provisioned with firewall functionality to enhance the security and isolation of multi-tenant deployments in which multiple logical servers share underlying physical servers. *See* MS-1006, 2:1-7, 8:39-64, 10:9-31; MS-1009, 17:16-19, 32:7-11, 32:13-33:10.

130. Hirschfeld provides a virtualization environment in which multiple logical servers are hosted on shared physical servers, where each logical server

“operates substantially independent of other logical servers and provides virtual isolation among users.” MS-1006, 2:1-7. Hirschfeld also recognizes that the shared network “may include a firewall or the like associated with a particular network fabric, which provides an interface to the network system.” MS-1006, 8:39-64. Hirschfeld provides a multi-tenant virtualization architecture that contemplates both virtual isolation among logical servers and network-level security through firewalls, but does not specifically describe implementing firewall functionality as a software application running within a logical server. MS-1006, 2:1-7, 8:39-64.

131. A POSITA seeking to implement Hirschfeld’s multi-tenant logical server cloud would have recognized the benefits of enhancing security and network isolation of the provisioned logical servers, particularly in environments where multiple users or organizations share the underlying physical servers over a network. Hirschfeld explicitly contemplates these shared multi-tenant environments and emphasizes the importance of virtual isolation among logical servers. MS-1006, 2:1-7. Tremain identifies that, in conventional hosting deployments, having “multiple organisations sharing a single applications instance running on a single real server” or “multiple organisations, each with their own applications instance running on a single real server” “has associated security, manageability and availability issues.” MS-1009, 4:6-19. Tremain further describes that it is highly desirable to provide "enhanced security isolation" to ensure that

virtual servers and their data are "held isolated from each other." MS-1009, 27:1-26.

132. In seeking techniques that address these security and isolation issues in a virtualized environment, a POSITA would have found Tremain, which, like Hirschfeld, focuses on hosting deployments in which multiple customers share physical server resources. MS-1009, 4:6-19. Tremain describes security features for these configurations by providing "embedded 'virtual security firewalls'" designed to "provide protection to virtual servers and virtual networks." MS-1009, 32:7-11. These virtual security firewalls "consist of a special firewall software application which runs on a host operating system on a computer" and are "preferably implemented by a virtual machine on the real computer... running firewall software." MS-1009, 32:13-33:10, 17:16-19.

133. A POSITA would have viewed Tremain's software-based virtual firewalls as a compatible enhancement to Hirschfeld's logical server cloud because both references contemplate hosting multiple virtual machines with independent operating systems and software applications on shared physical servers to provide services to users over a network. *Compare* MS-1006, 10:9-31 (logical servers with independent operating systems and software layers hosted on shared physical servers), *and* MS-1009, 17:16-19 (providing computer services by setting up virtual machines running specific software applications, including firewalls).

134. To incorporate Tremain's software-based virtual firewalls within Hirschfeld's system, a POSITA would have made the firewall software available as an application program within Hirschfeld's logical server cloud, so that logical servers may be provisioned with firewall functionality through the SCM's existing management and instancing procedures. *See* MS-1006, 10:9-31, 15:23-67; MS-1009, 32:13-33:10. This configuration advances Tremain's stated objective of providing enhanced security isolation and protecting virtual networks from unauthorized access, while remaining consistent with Hirschfeld's architecture in which the SCM manages the provisioning and association of logical servers (each executing independent operating systems and desired software applications) to physical servers. MS-1006, 2:1-7, 10:9-31; MS-1009, 32:13-33:10.

135. A POSITA would have also reasonably expected to succeed in combining the teachings of Hirschfeld and Tremain in the manner discussed above. Both references address virtualized computing environments in which software applications are deployed onto virtual machines hosted on physical servers. MS-1006, 10:9-31; MS-1009, Abst. The combination involves applying Tremain's teaching of a software-based virtual firewall as a predictable application program within Hirschfeld's logical server cloud, which already supports logical servers executing independent operating systems and desired software applications. MS-1006, 10:9-31; MS-1009, 32:13-33:10. A POSITA would have understood this as

combining known elements (virtual machines, application software, and firewall software) according to known methods (provisioning logical servers with specific software applications) to yield predictable results (a securely provisioned logical server).

E. The Combination Of Pabari And Bonazzola

136. The Pabari-Bonazzola System would have combined Pabari's virtualization management architecture with Bonazzola's techniques for local provisioning of virtual machines on host machines. In the combined system, Pabari's computer device (e.g., computer device 300) would be configured to execute a local provisioning component similar to Bonazzola's setup component 130. The Bonazzola-compatible setup component within computer device 300 would have been configured to receive user-specified configuration parameters (MS-1013, [0022], [0023]), configure storage and networking (*id.*, [0024]), create and deploy a VM on the host machine (*id.*, [0024], [0025]), and register the provisioned VM with a management system (*id.*, [0026]). This local provisioning component would execute on the processor of the computer device (e.g., CPU 306) alongside Pabari's Control Agent 324 and virtualization layer 322. This configuration would have provided provisioning-related operations being performed locally by the host device's processor, rather than exclusively by the Control Center. See MS-1001, 9:39-61, 10:18-42, FIG. 3; MS-1013, [0011],

[0021]-[0026], FIGS. 2-3.¹

137. Pabari describes a Control Center application that manages provisioning of virtual assets across host machines through a Provisioning Manager 206 and other functional components (elements of the Control Agent 324 separate from the computer device 300). MS-1001, 8:17-40, FIG. 2. However, Pabari does not describe or suggest how the processor of the computer device (e.g., CPU 306 of computer device 300) is itself configured to receive provisioning requests specifying configuration parameters, locally execute a provisioning workflow to deploy a virtual asset matching those parameters, associate the provisioned asset with the originating request, or independently provide user access to the provisioned asset. See MS-1001, 8:17-40, 10:18-42, FIG. 3. In Pabari's architecture, these provisioning operations are attributable to the Control Agent 324 (not to the CPU 306 of computer device 300).

138. Bonazzola describes a virtualization system in which a setup

¹ For simplicity, my analysis cites to MS-1001 (the '050 patent specification) when addressing the disclosure of Pabari because the two share substantively identical disclosure contents. Pabari is a publication in the priority chain of the '050 patent and contains the same specification text. Citations to MS-1001 in the context of this section therefore correspond to the same disclosure as would appear in Pabari.

component 130, executing on a host machine's processor (e.g., processing device 402), locally performs provisioning operations. MS-1013, [0011], [0021]-[0026], FIGS. 2-3. The setup component 130 receives configuration information from a user. This configuration information includes storage configuration (e.g., shared storage connection path, storage domain, datacenter name), network configuration (e.g., NIC information), VM configuration (e.g., memory size, number of virtual CPUs), and management system configuration (e.g., hostname, password). The setup component 130 uses this information to locally set up network connectivity, prepare storage locations for virtual disks, create and start a VM on the host machine, install an operating system, and register the host machine with a management system. *Id.*, [0022]-[0026]. The setup component 130 also "performs the noted activities automatically and without user involvement." *Id.*, [0024].

Beyond this specific use case, a POSITA would have understood these local provisioning techniques to be applicable to provisioning of virtual assets generally, because Bonazzola's management system 140 is described as capable of "provisioning new VMs" and "creat[ing] virtual machines" using "VM templates that are used for creating new VMs with predefined settings and preinstalled disk images." *Id.*, [0020].

139. A POSITA implementing Pabari's virtualization management architecture would have been motivated to incorporate local provisioning

capabilities that enable the processor of a host device to locally execute provisioning operations that Pabari attributes to the Control Center. This is because Pabari's architecture already contemplates host devices with processors, virtualization layers, and Control Agents that facilitate communication between the host and the Control Center. See MS-1001, 9:39-61, 10:18-42). This creates an integration point for a local provisioning component. Bonazzola also identifies a benefit of local provisioning on host devices, as its approach “improve[s] virtualization systems by allowing the management system to be deployed using less hardware than previous implementations (as the management system is not utilizing a separate machine to execute).” MS-1013, [0013]. This configuration would have also reduced reliance on the Control Center for provisioning workflows, improved provisioning latency by enabling local execution on the host, and enabled provisioning in environments where connectivity to the Control Center is intermittent or unavailable.

140. A POSITA would have found Bonazzola’s setup component 130 (which receives configuration parameters and locally provisions a VM on the host machine’s processor) to be relevant to Pabari’s architecture. Incorporating those teachings into the system would have been a predictable, compatible refinement to Pabari's computer device, because it supplies a known mechanism for performing provisioning operations locally on a host device (operations that Pabari’s Control

Center already orchestrates remotely). MS-1001, 8:17-40; MS-1013, [0021]-[0026]. Pabari's agent-based architecture (in which the Control Agent 324 already communicates with the Control Center to manage the host's virtualization layer) also supports Bonazzola's local provisioning component, because the computer device's processor already executes software that interfaces with both the virtualization layer and the Control Center. MS-1001, 10:18-42.

141. In one example implementation, Pabari's computer device 300 is configured to execute a Bonazzola-compatible setup component (e.g., setup component 130) on its processor (e.g., CPU 306). In this example, the setup component receives configuration parameters specifying how a VM should be provisioned on the host, generates installation settings based on those parameters, and locally provisions the VM by configuring storage, networking, and compute resources on the host machine. MS-1013, [0022]-[0024], [0030], FIGS. 2-3. After provisioning, the setup component registers the provisioned VM with the management infrastructure (e.g., by sending a command to add the host to a list of managed hosts and waiting for confirmation) and provides the user a temporary password for direct access to the VM. *Id.*, [0025], [0026]. The Control Center would continue to provide centralized management functions (e.g., virtual mapping, capacity planning, performance management) as described in Pabari, while the host device's processor would additionally be capable of locally

executing provisioning workflows. MS-1001, 7:33-56, 8:17-40.

142. Incorporating Bonazzola's teachings would have reflected an implementation for enabling host-local execution of provisioning operations within Pabari's existing virtualization management architecture, while preserving Pabari's Control Center management functions and overall system design. By enabling the processor of the computer device to locally perform the provisioning-related operations, the combination would have advanced Pabari's overall objective of providing automated management of virtualization environments across computer networks. See MS-1001, 3:45-65. A POSITA would have also reasonably expected success in doing so because the combination amounts to applying a known host-level provisioning technique (Bonazzola's setup component executing on the host machine's processor) to a known virtualization management architecture (Pabari's Control Center and host device arrangement) that already operates through well-defined agent-based interfaces and already contemplates host devices with processors executing virtualization software.

X. THE PRIOR ART ANTICIPATES CERTAIN CLAIMS

A. Ground 1A: Claims 1, 3, 4, 6-8 Are Anticipated By Khandekar

1. Claim 1

[1.pre] A method for on demand provisioning of virtual asset via a self-service portal in a virtualization system, the method comprising:

143. If the preamble is treated as limiting, Khandekar describes an on-

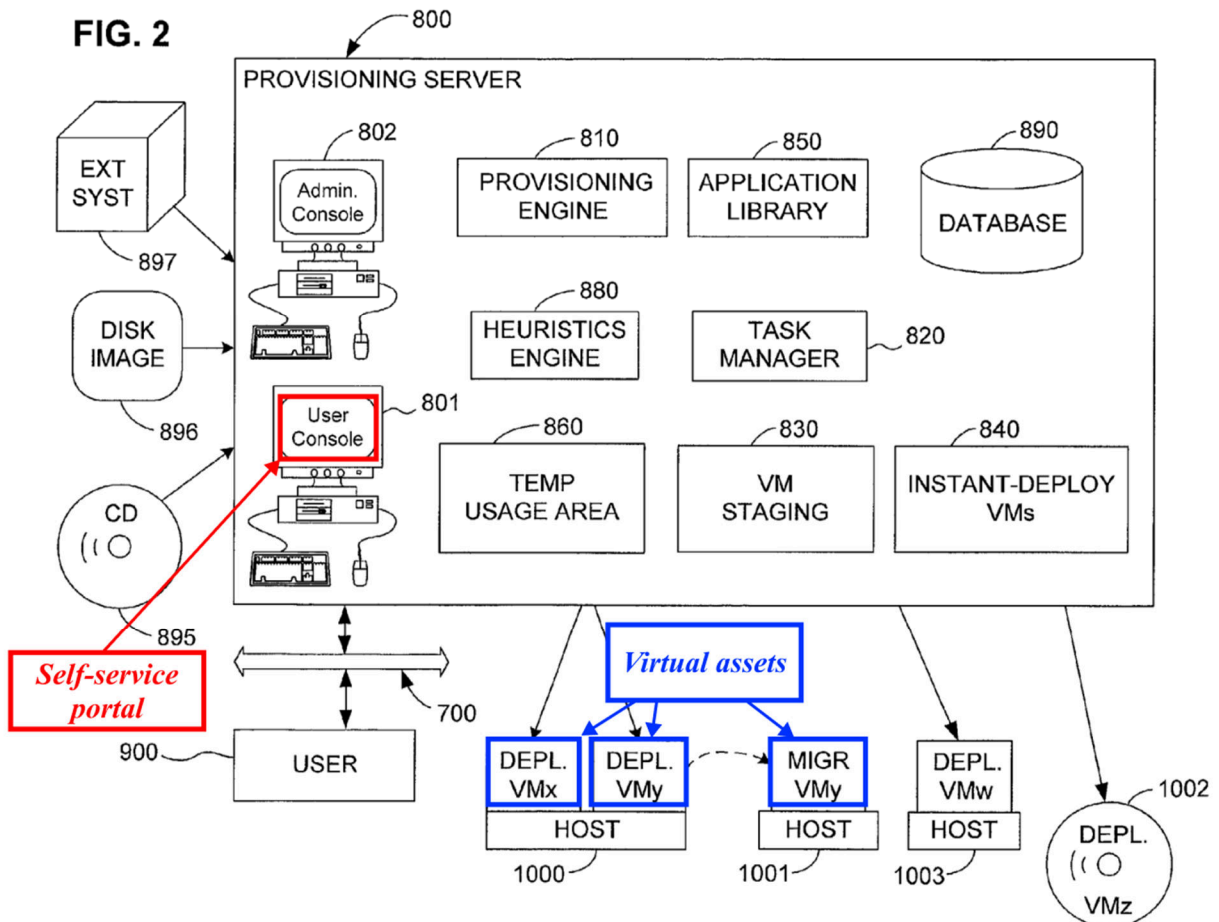
demand workflow where a user requests a virtual machine that exists only in a virtualization environment through a browser-accessed console and the system provisions that VM automatically. I refer to the general system described in Khandekar as the “Khandekar System” when convenient in what follows. In Khandekar, the “virtual asset” is a VM that is configured and deployed by the provisioning components without manual administrator steps once the request is made. The user makes the request through the user console (e.g., 801), which Khandekar describes as preferably operating as a web server accessed using a conventional web browser—i.e., a self-service portal. The provisioning occurs in the disclosed provisioning server/host environment (e.g., provisioning server 800 interacting with hosts 1000/1001), which is a virtualization-based provisioning system of the type addressed by the claim. MS-1004, 1:12-14, 1:64-2:14, 5:25-28, 9:20-25, 9:46-64, 11:20-35, 13:66-14:6, 15:29-39, 21:26-31, 22:35-51; *see* MS-1001, 8:17-40.

144. Khandekar’s stated objective is to reduce the manual effort associated with configuring a new computer and to enable deployment quickly through automation. From a POSITA’s perspective, that is an “on demand” provisioning model because the provisioning engine begins execution as a direct response to the user’s request and then carries the steps through to completion without requiring a human operator to perform the intermediate configuration tasks. MS-1004, 1:12-

14, 1:64-2:14, 5:25-28, 9:20-25, 11:20-35, 15:29-39, 21:26-31. Specifically, Khandekar's technical disclosures focus on mitigating the manual “frustrations involved in configuring a new computer.” It describes overcoming prior art processes that are “difficult to automate completely” by offering a provisioning mechanism that “takes only a few seconds and the user can start using it almost instantaneously.” MS-1004, 1:12-14, 1:64-2:14, 11:20-35. A POSITA would recognize this automated workflow as on demand provisioning because the Khandekar System is designed to operate “with no need for human intervention.” *Id.*, 5:25-28. These automated workflows, including Khandekar's “instant-deploy” and automated “custom-built” routines, represent examples of on demand operations. *Id.*, 9:20-25, 15:29-39, 21:26-31.

145. With respect to the “self-service portal” aspect, Khandekar explains that the user interacts with the provisioning server via the user console using a conventional web browser, and that through this interface the user can select a pre-configured VM or initiate custom provisioning. A POSITA would understand this as a self-service web portal because it is the mechanism that exposes provisioning capabilities directly to the user in a controlled way, while the server-side system performs the provisioning actions across the host environment. Khandekar explains that the user interface is accessed through a user console (e.g., 801), which is “preferably a web server, with which the user interacts with any conventional web

browser.” MS-1004, 13:66-14:6. Through this interface a user “logs in to the provisioning server” and subsequently “can select any pre-configured” VM or choose to “custom-provision” a VM, which initiates provisioning operations. *Id.* A POSITA would understand that this type of user portal allows users to self-serve their own computing resource needs while working in the networked provisioning server and host environment. *Id.*, 9:46-10:3, 22:35-51. The annotated version of Figure 2 below illustrates this architectural arrangement.



MS-1004, FIG. 2 (annotated)

[1.1] providing a virtualization system that has an inventory that includes physical resources and virtual assets managed by a virtualization layer running on the physical resources;

146. Khandekar discloses a system providing a hosted computing architecture in which a provisioning server 800 interacts with multiple host machines (e.g., 1000 and 1001) to deploy and manage virtual machines. This corresponds to the recited virtualization system, a virtualization-based provisioning environment of the type described in claim 1. The system includes a database (e.g., 890) tracks resources of hosts (*physical resources*) and VMs (e.g., “Instantly deployable VMs” or “Staging VMs”) (*virtual assets*) that collectively represent *inventory*. Khandekar further explains that these resources are controlled through a virtual machine monitor (“VMM”), which a POSITA would understand as the virtualization layer executing on the hosts (which serve as the *physical resources*). MS-1004, 7:7-10, 7:28-38, 8:20-35, 9:26-32, 10:27-42, 13:17-29, 18:2-35.

147. A POSITA would have understood that the term “inventory” includes “resources and assets in the virtualization system and information thereof.” For example, the ’050 patent specification describes “building an inventory of available physical resources and virtual assets” as the first step in the process of managing the virtual environment. MS-1001, 4:1-10. This description indicates examples of “inventory” involves a process of collecting and organizing information about resources and assets. Additionally, “an inventory of resources

and assets available at a network virtualization environment is determined” and that “prioritization is assigned to an inventory of available resources and assets.” *Id.*, 4:11-35. Determining an inventory and assigning prioritization to it are operations performed on information about the resources and assets.

148. Khandekar explains that a provisioning server (e.g., 800) coordinates provisioning across multiple hosts. MS-1004, 9:26-32, 10:27-42. Khandekar also describes a database (e.g., 890) that serves as a “central repository of data required by the various components.” *Id.*, 18:2-5. The repository includes data structures identifying the available both hosts (representing physical resources) and “Staging VMs” or “Instantly deployable VMs” (representing virtual assets). *Id.*, 18:6-42. In addition, Khandekar states that the database also keeps “information about the hosts available” and indexes the “VM configuration” for virtual assets. *Id.*

149. Khandekar further describes that a “virtual machine monitor (VMM), is installed on each of a plurality of hardware hosts.” *Id.*, 4:66-5:2. Khandekar explains that the VMM “virtualizes all, or at least some of, the resources of the machine” which is how a POSITA would understand the virtualization layer to operate in a virtualized hosting environment *Id.*, 7:7-10. The VMM manages VMs by performing resource planning through the identification and management of network resources and virtual assets, e.g., by “schedul[ing] and handl[ing] all requests by its VM for machine resources.” *Id.*, 7:24-38.

150. Khandekar also describes the VMM as carrying out these management functions in a way without regard to processors, operating systems, virtualization platforms, and application software. In particular, Khandekar explains that the VMM’s resource mapping “can be accomplished in a manner that is transparent to the VM, and that does not require modifications to either the guest OS...or to the underlying host operating system.” *Id.*, 8:23-36. This allows VMs to migrate freely “even to other hardware and software platforms.” *Id.* Khandekar further notes that the VMM is executed on hardware since it is “usually a thin piece of software that runs directly on top of the host.” *Id.* In this disclosed architecture, the hosts (*physical resources*) run a VMM (virtualization layer) to support the virtual machines (*virtual assets*) that are tracked and maintained in the system’s database (*inventory*).

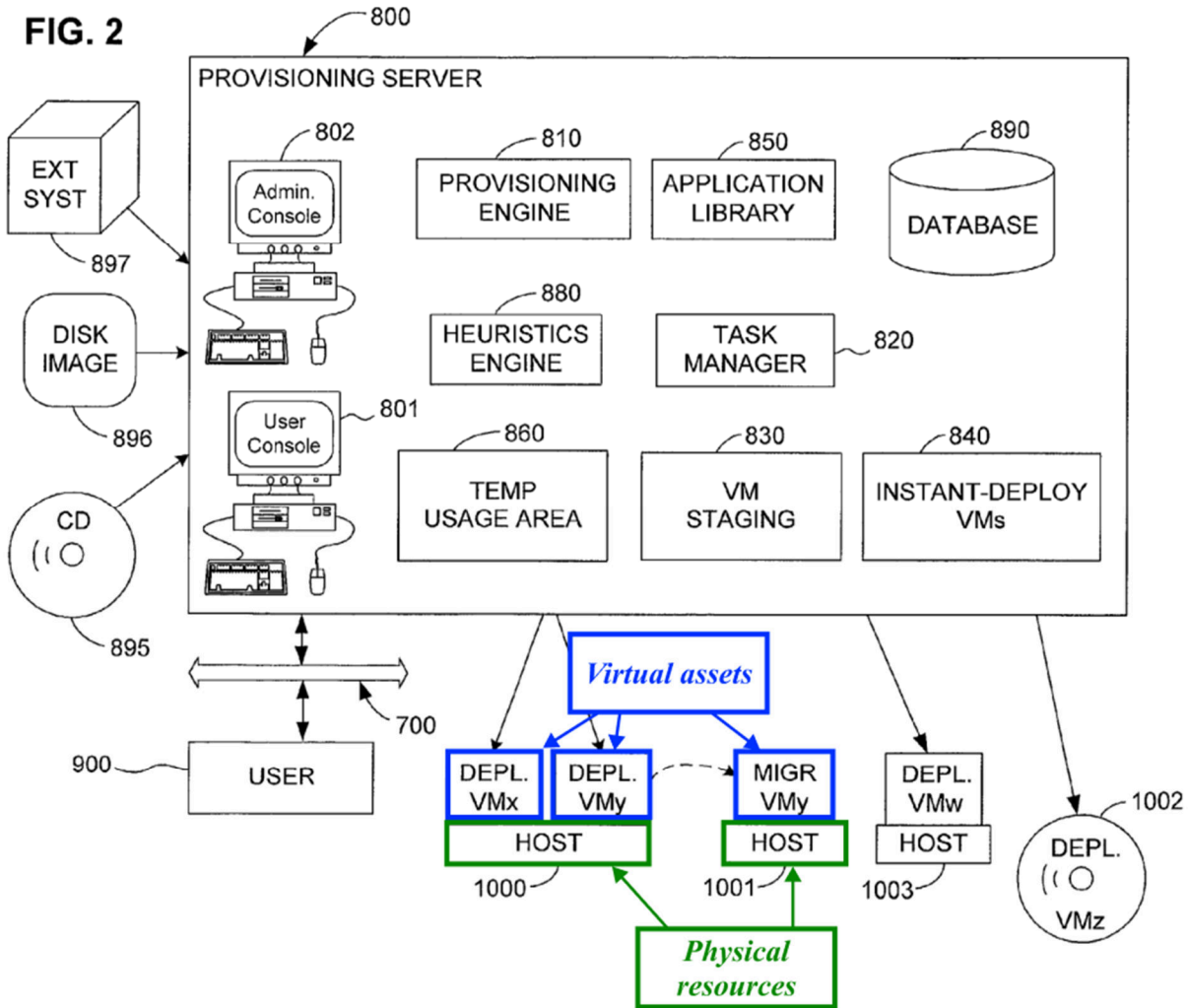
[1.2] wherein virtual assets are software instantiations of computer systems;

151. In the Khandekar system, “Virtual Machines” or “VMs” (which are *virtual assets*) include software that can virtually perform functions of physical devices that can connect to computers (e.g., “software abstractions” or “software constructs”) (*software instantiations of computer systems*). MS-1004, 6:24-61, 7:44-47.

152. As I discussed for limitation [1.1], Khandekar describes a “virtual machine (VM)” as “a software abstraction—a ‘virtualization’—of an actual

physical computer system.” MS-1004, 6:24-27. In Khandekar, VMs include software constructs that emulate connected physical hardware, such as “optional virtual peripheral devices 340 (VDEVICES)” that are “implemented in software to emulate the corresponding components of an actual computer.” *Id.*, 6:32-40.

153. As another example, the system includes “device emulators 540 that emulate the characteristics of respective physical devices.” *Id.*, 7:48-55. A POSITA would understand these “device emulators” as software components that replicate, in software, the behavior of corresponding physical devices. Because these emulators provide software-based implementations of device characteristics and functions that would otherwise be provided by physical computer hardware, they show the Khandekar VMs being implemented as software constructs. The VMs provided by the Khandekar System therefore include software that are examples of *software instantiations of computer systems*.



MS-1004, FIG. 2 (annotated)

[1.3] receiving a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request;

154. In Khandekar, a user submits a provisioning request through the user console (e.g., 801). That submission initiates the process of provisioning a VM and therefore is an example of a virtual-asset provisioning request. The request can include user-selected hardware characteristics (e.g., memory and disk selections) as well as operating-system configuration information (e.g., hostname and IP

address), which Khandekar identifies as inputs, or parameters, used in provisioning the VM. MS-1004, 9:46-10:14, 14:31-64, 20:64-67.

155. As I discussed for limitation [1.pre], in the Khandekar System, a user interacts with the user console 801 to initiate provisioning. MS-1004, 9:46-64. For example, Khandekar states that “[w]hen the user submits a request to provision a new VM, the provisioning engine requests that a new task be created.” Id., 20:64-67. This request includes specific configuration details explicitly identified in Khandekar as parameters. For example, for custom-built VMs, the “user specifies the various (virtual) hardware components” and “[t]he various parameters that the user will normally need to specify” include “[t]he main memory (RAM),” the “number of disks,” and the “size of each disk.” Id., 14:31-64. Additionally, the user specifies the “desired OS configuration, for example, hostname, IP address.” Id., 9:65-10:14. A POSITA would have understood that these user-selected hardware and OS inputs constitute “parameters to be considered” because they are the specific configuration details the provisioning engine uses to determine how the VM will be built, configured, and deployed.

[1.4] receiving data on inventory available in the virtualization system;

156. The Khandekar System receives “information about the current state” of hosts and VMs (which serves as data on inventory) available in the provisioning server environment (representing availability in the virtualization system). MS-

1004, 13:1-4, 18:33-39, 20:54-62.

157. As I discussed for limitation [1.1], in Khandekar, the system maintains inventory information in a database (e.g., Database 890). This includes “[i]nformation about the current state of the different hosts is preferably input dynamically into a database or other data structure in the heuristics engine 880.” MS-1004, 12:67-13:3. Khandekar also describes maintaining “information about the hosts available to deploy VMs on,” such as “[h]ardware configuration of the host (memory size, disk space, etc.).” Id., 18:33-39.

158. Khandekar further describes how the system obtains this inventory information from the hosts. For example, the heuristics engine 880 can “[s]can the list of available host machines 1000, 1001, etc.,” and “determine the one with enough memory and disk space” by “querying the host via an API provided to manage VMs.” Id., 20:54-58. The host then responds by “[r]eturn[ing] the configuration of the host” back to the provisioning server. Id., 20:61-62. This exchange—querying hosts through a VM-management API and receiving each host’s current configuration in response—demonstrates the Khandekar System (and specifically the provisioning server) receiving up-to-date data about the available inventory in the provisioning environment.

[1.5] automatically provisioning of a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request;

159. The Khandekar System automatically configures and deploys a specific VM instance (e.g., VMx) (*particular virtual asset*) to a selected host (e.g., 1000) (*particular physical resource*) in the provisioning environment (*virtualization system*) without human intervention (*automatically provisioning*). In doing so, the system can take into account requirements specified by the user in the provisioning request (e.g., operating system, desired RAM) (*parameters in the virtual asset provisioning request*), and the system can provision the VM to match the specified parameters (*match[ed]*) (e.g., VM having a specific OS with the amount of RAM as the desired RAM). MS-1004, 14:26-49.

160. As I discussed for limitations [1.3], Khandekar explains that, for custom-built VMs, the user provides information about the “various (virtual) hardware components of a VM in order for the provisioning server to be able to create the VM.” MS-1004, 14:31-36. Khandekar gives examples of such request parameters, including “OS selection” (*id.*, 14:26-29) and a desired amount of “main memory (RAM)” (*id.*, 14:31-49). The Khandekar System uses those user-specified inputs when constructing the VM image/instance, so that the provisioned VM matches the requested configuration, e.g., the same OS and the “desired amount of memory.” *Id.* This requires provisioning the VM to a suitable physical resource. In this way, the Khandekar System *match[es]*, for example, the OS and RAM amount of the VM to the specific types specified in the provisioning request

(e.g., selected OS, desired amount of RAM).

[1.6] assigning the provisioned virtual asset to the virtual asset provisioning request; and

161. The Khandekar System associates (assign[s]) a deployed VM (provisioned virtual asset) to a specific transaction task and temporary usage area generated for a user-submitted request (virtual asset provisioning request). This reflects an example of assigning the provisional virtual asset to the virtual asset provisioning request. MS-1004, 10:61-67, 15:29-39, 19:20-25, 20:64-21:8, 21:39-42, 21:50.

162. As I discussed for limitation [1.3], the provisioning process in the Khandekar System is initiated by a user request. Once the request is received, “a new transaction is started in the task manager 820 and the user is returned a task ID.” MS-1004, 15:29-39; see 10:61-67. Khandekar further indicates that this task ID “identifies the task” associated with the specific provisioning request. Id., 20:64-21:1-8. To carry out the request, Khandekar also describes creating “a temporary usage area 860 for the provisioning request that holds the user input as well as the various temporary data needed while the provisioning task is in progress.” Id., 10:61-67.

163. Khandekar adds that “[f]or each VM in the process of being provisioned, a corresponding storage space 866 is preferably provided” within this temporary usage area. Id., 19:20-25. Khandekar also describes steps that the

provisioning server performs to map the requested VM to this, such as “[c]reat[ing] a new temporary usage area 860 for this task,” storing the user’s configuration choices, and creating the necessary files to “[c]opy the VM’s config file and the disk files to the host.” Id., 21:39-42, 21:50. Accordingly, by linking the creation and deployment of the specific VM to the dedicated temporary usage area and task ID generated for the user's submission, the Khandekar System assigns the provisioned VM (the provisioned virtual asset) to the user-submitted request (the virtual asset provisioning request).

[1.7] providing user access to the provisioned virtual asset.

164. The Khandekar System enables a user to access provisioned VMs (representing provisioned virtual assets) deployed on and running on the hosts through the interfaces disclosed for user interaction, including the user console and remote-access mechanisms. This is an example of providing user access to the provisioned virtual assets. MS-1004, 9:65-10:14, 13:66-14:18, 15:29-39, 20:64-21:8, 21:9-15. For example, a user may use “remote access software” or “remote console software” to connect to a provisioned VM. MS-1004, 9:65-10:14, 13:66-14:18, 15:29-39, 20:64-21:8, 21:9-15.

165. Khandekar explains that the user connects to the provisioning server via the user console, which it describes as “preferably a web server...” MS-1004, 13:66-14:3. Through this connection, the user can, for example, access VM-related

resources by selecting a pre-configured VM (e.g., an “instantly deployable” VM) or submitting configuration selections (e.g., OS configurations such as hostname, IP address, and virtual hardware parameters) for a custom-provisioned VM. *Id.*; *see id.*, 9:65-10:14, 14:4-18. Khandekar further explains that after the request is submitted the system returns a task identifier, and the user can use the console/browser connection to check task status and, in some cases, abort the provisioning task before it completes. *Id.*, 15:29-39, 20:64-21:8, 21:9-15.

166. Khandekar also describes how the user is notified when provisioning is complete (e.g., “via the browser from which he submitted the request”). *Id.*, 15:29-39. And, once the VM is deployed, Khandekar explains that “[t]he user can then access the newly created VM via any known remote access software,” e.g., “...software available from VMware.” *Id.*, 14:4-18. A POSITA would understand these disclosures as providing the user who submitted the request with the practical ability to connect to and use the provisioned VM, which is an example of providing user access to the provisioned virtual asset.

2. *Claim 3*

[3] *The method of claim 1 further comprising providing a user interface for submitting the virtual asset provisioning request.*

167. The Khandekar System provides a user console (e.g., 801) that presents “dialogs” or web pages (which function as a user interface) to the user for “submit[ting] the selected options” to a provisioning server (corresponding to

submitting the virtual asset provisioning request). MS-1004, 10:15-26, 13:66-14:3, 14:26-29, 15:16-28.

168. As I discussed for limitation [1.3], the user console that is used to provide provisioning requests “is preferably a web server, with which the user interacts with any conventional web browser.” MS-1004, 13:66-14:3. To facilitate the request, Khandekar explains that “[a]ny conventional interface may be used in the user console 801 to present the various options to the user and to submit the selected options to the provisioning server.” *Id.*, 10:15-26. Furthermore, “the console 801 preferably presents three primary dialogs for the user,” including a “Hardware Selection Dialog,” “OS Selection Dialog,” and “Application Selection Dialog.” *Id.*, 14:26-29, 15:16-28. A POSITA would recognize these web-based, browser-accessed dialogs as standard examples of a user interface used to gather selections and submit a provisioning request to the server.

3. *Claim 4*

[4] *The method of claim 3, wherein providing the user interface further comprises providing a user interface to access one or more images in a virtual asset image repository.*

169. The Khandekar System includes a user console 801 (*user interface*) that can present the user with a list of “staged VMs” or “instantly deployable VMs” whose underlying configurations are associated with image files (e.g., “disk image,” “ISO image,” “suspend-to-disk image file”) (corresponding to one or more

images). Khandekar also describes storing such VM resources and images in designated repositories or libraries (e.g., repository 840, “library of disk drive images”) (representing a virtual asset image repository) for user selection (which corresponds to access). MS-1004, 10:15-42, 13:66-14:3, 15:55-65, 17:31-59, 18:2-15.

170. As I discussed for claim 3, Khandekar explains that the user console used to provide provisioning requests “is preferably a web server, with which the user interacts with any conventional web browser.” MS-1004, 13:66-14:3. Through this interface, Khandekar explains that “a list of available staged VMs” can be presented for display” and that “the user may then select one of the VMs from the list.” *Id.*, 10:15-26. These VMs are stored in designated repositories, such as a “library 830 of staged VMs” or an “instantly-deployable VMs repository 840.” *Id.*, 10:27-42, 18:14-15. Khandekar also associates deployable VM configurations to image files. For example, instant-deploy VMs stored in repository 840 include a “suspend-to-disk image file 844.” *Id.*, 17:31-49. Users may also create a model VM from a “library of disk drive images” or install an OS from an “ISO image.” *Id.*, 15:55-67. A POSITA would understand that presenting these selectable VM options through the browser-accessed console, where the underlying images are stored in the disclosed repositories/libraries, is an example of providing a user interface to access one or more images in a virtual asset image repository.

4. *Claim 6*

[6] The method of claim 1, wherein the inventory further comprises one or more application programs.

171. The Khandekar System maintains information within database 890 (e.g., state information) (*inventory*) that tracks and includes data structures for “Applications” available for installation from a library (corresponding to *one or more application programs*). MS-1004, 18:24-26, 15:23-28, 18:57-19:2. A POSITA would understand that this is an example of an inventory that includes one or more application programs.

172. As I discussed for limitation [1.1], the Khandekar System maintains an inventory using a database (e.g., 890). This database includes information, such as “Applications,” which is a “data structure [that] keeps information about the applications available for installation in a custom-built VM.” MS-1004, 18:24-26. This inventory data structure indexes an “Applications Library 850,” which is “a repository of the software and data needed to install an application 856 in a VM.” *Id.*, 18:57-19:2. Khandekar also describes presenting an “Application Selection Dialog” through the user console 801, where the user “selects the applications he wants installed in the VM.” *Id.*, 15:23-28. A POSITA would understand that in this arrangement, the system’s inventory database tracks available application software (programs), and additionally includes examples of one or more application programs.

5. *Claim 7*

[7] The method of claim 6, wherein one of the application programs comprises one of a network load balancer, a router, a firewall and a storage server.

173. The Khandekar System includes an “Applications Library” (*application programs*) with code for “database servers” (*storage server*). MS-1004, 11:50-58, 23:11-20. Khandekar describes that a VM may run an application representing an “database server” thereby enabling access to a storage server. MS-1004, 11:50-58; *see id.*, 23:11-20. Because of the phrase “one of,” satisfaction of claim 5 does not mandate providing end user access to each of the enumerated elements recited in claim 7.

6. *Claim 8*

[8] The method of claim 6, wherein one of the application programs comprises a network server.

174. The Khandekar System includes an “Applications Library” (representing *application programs*) with code for “web servers” or “database servers,” which are applications that serve data over a network (corresponding to a *network server*). MS-1004, 11:50-58, 23:11-20. In particular, a POSITA would understand those types of applications as network-facing server software that receives requests and provides data or services over a network, i.e., examples of a network server.

175. As I discussed for claim 6, the system inventory includes an

“Applications Library 850” containing various software applications. Khandekar explains that this library “could contain code for any collection of the myriad of existing applications, from web servers, database servers, spreadsheets and word-processing programs, to games and browsers.” MS-1004, 11:50-58. Additionally, a network of computers may be provisioned where “a developer wants to test his web application, which uses a web server on one computer, an application server on another, and a database server on yet another.” *Id.*, 23:11-20. A POSITA would have understood that “web servers” and “database servers” are applications designed to process requests and deliver data across a network (e.g., providing web content or database query results to other machines), and thus represent examples of a network server under its plain and ordinary meaning.

XI. THE PRIOR ART MAKES CERTAIN CLAIMS OBVIOUS

A. Ground 1B: Claims 9, 11, 12, 14, And 16 Are Obvious Over Khandekar

176. A POSITA would have found it obvious that Khandekar’s provisioning server 800 is configured to operate as a centralized management apparatus in which the heuristics engine 880 and database 890 collectively function as a virtual mapping layer by supporting the orchestration of VM provisioning. This is supported by Khandekar’s disclosure, which explains that “[t]he provisioning engine is the component that ties together the various components described above.” MS-1004, 21:26-31. Khandekar also states that “[t]he database

890...is a central repository of data required by the various components of the provisioning server.” *Id.*, 18:2-5. Additionally, “the provisioning server passes this information to the heuristics/rules engine 880, which then matches the information against the database 890 to select the appropriate host and the VM hardware configuration.” *Id.*, 20:32-42. These descriptions collectively suggest that the configuration of the provisioning server 800 operating as a centralized management apparatus (and in which the heuristics engine 880 and database 890 collectively support the orchestration of VM provisioning) is consistent with improvements provided by the invention disclosed in Khandekar.

177. A POSITA would have also recognized that the integrated, centralized configuration discussed above (provisioning server 800 operating with provisioning engine 810, heuristics engine 880, and database 890 as coordinated components) is an available implementation reflected in Khandekar. *See* MS-1004, 9:26-45, 21:26-34. This is consistent with Khandekar, which describes that the provisioning engine 810 “call[s] the heuristics engine 880, to determine the VM's hardware configuration and the host to deploy on,” and that the provisioning server passes user information to the heuristics/rules engine 880, which matches that information “against the database 890” to select the appropriate host and VM hardware configuration. *Id.*, 20:32-42, 21:26-34. A POSITA would have understood these interactions among provisioning engine 810, heuristics engine

880, and database 890 as reflecting a coordinated provisioning workflow within provisioning server 800 for determining where and how VMs are allocated to available host resources.

178. A POSITA would have also found it obvious that, when configured to operate as discussed above, the heuristics engine 880 and database 890 determine inventory irrespective of any specific processors, operating systems, virtualization platforms, or application software. Khandekar's database 890 catalogs VMs across multiple operating systems (e.g., "Linux Red Hat, Windows 2000, Windows Me") and a broad range of applications, e.g., "web servers, database servers, spreadsheets and word-processing programs, to games and browsers." MS-1004, 11:36-58. A POSITA would have understood that by cataloging VMs with diverse operating systems and application configurations as available options within the same system, Khandekar's provisioning framework accommodates and supports provisioning across these different platforms. The provisioning environment is also designed to maintain inventory across heterogeneous operating system environments and heterogeneous application stacks. When the database is structured to track VMs and applications across that breadth of configurations, the inventory function operates at the management level across differing software environments rather than being constrained by any particular one of them.

179. Khandekar also describes host selection in terms of generalized

resource availability rather than platform-specific identity. Khandekar explains that the heuristics engine scans “the list of available host machines” and determines “the one with enough memory and disk space to host the VM about to be created.” MS-1004, 20:54-58. This selection logic is directed to whether a host has sufficient available resources to satisfy the provisioning request, not whether the host corresponds to a particular processor family, operating system, or virtualization product label. When a management component chooses among candidate hosts by evaluating available memory and disk capacity, it is operating at an abstracted resource allocation level (rather than at the level of machine-specific implementation details). This supports the understanding that the inventory and selection functions of heuristics engine 880 and database 890 are applied irrespective of any particular processors, operating systems, virtualization platforms, or application software.

180. Khandekar also reinforces that understanding by describing VM deployment as “substantially independent of the actual physical hardware configuration” so long as the VMM on each host “exports a known hardware interface to the VM deployed on it.” MS-1004, 13:17-35. The resource mapping can be performed transparently to the VM, without modification to the guest OS or underlying host operating system, such that VMs may “migrate freely, even to other hardware and software platforms.” *Id.*, 8:20-35. This reflects a provisioning

architecture in which the management-side functions operate through a virtualization abstraction that masks underlying hardware and platform variation. In such an architecture, provisioning server 800 can determine available inventory and make allocation decisions irrespective of any (e.g., without being constrained) by the specific processors, operating systems, virtualization platforms, or application software on any given host. This is because the relevant management decision is made against abstracted resource and configuration information rather than against platform-specific implementation details.

1. *Claim 9*

[9.pre] An apparatus that performs automatic management of a virtualization environment in a computer network comprising:

181. The Khandekar System (*apparatus*) would have included a provisioning server 800 configured to “automatically configure[] and deploy[]” virtual machines (*perform[] automatic management*) and hosts 1000, 1001 connected via a network 700 (*virtualization environment in a computer network*). MS-1004, 5:25-36, 9:26-32, 10:50-60. The Khandekar System is an example of a system that includes two or more devices (e.g., server 800, host 1000).

182. As I discussed for limitation [1.pre] in Ground 1A, the Khandekar System enables the “provisioning of computer systems using virtual machines.” MS-1004. The system includes, for example, a “VM provisioning server 800” that is “configured as a network server” and implemented using “conventional network

server technology.” *Id.*, 9:26-32. This server automatically manages the environment by creating and deploying virtual machines “with no need for human intervention.” *Id.*, 5:25-36. As shown in the annotated FIG. 2 below, provisioning server 800 is connected via a “network 700” to multiple “host” computers (e.g., 1000, 1001). *Id.*, 10:50-60. A POSITA would have understood that the Khandekar System provides a similar networked arrangement of a management server and hosts that support virtual machines, where the networked management server coordinates deployment of VMs onto host machine, as an example of a virtualization environment in a computer network.

[9.1] a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device,

183. The Khandekar System would have provided a provisioning environment (*virtualization environment*) having hosts 1000, 1001 that include a CPU and RAM (*computer device with a processor and memory*), and a provisioning server 800 (*control center computer*) connected to the hosts via network 700 (*coupled to the computer device*). MS-1004, 5:59-67, 10:50-60.

184. Khandekar describes deploying VMs onto “host” computers, such as hosts 1000 and 1001. MS-1004, 10:50-60. Khandekar further describes that the underlying hardware platform for such hosts includes “one or more central processors CPU(s) 110” and “system memory 112, typically implemented using

any of the various RAM technologies.” *Id.*, 5:59-67. Based on this description, each host is an example of a ***computer device with a processor and memory*** because Khandekar identifies the host as the hardware platform on which the VM is deployed and describes the host hardware as including CPU and RAM components. *Id.* These hosts are part of the ***virtualization environment*** because they are the machines on which the virtual machines are instantiated and run. *Id.*, 10:50-60.

185. Khandekar also describes that the provisioning server is “configured as a network server” and communicates with the hosts over the network. MS-1004, 9:33-45, 10:50-60. Khandekar further describes that the provisioning server selects “which host machine 1000, 1001 is most appropriate to provision and deploy the VM on” and coordinates deployment of the VM to that selected host. *Id.*, 10:27-42. From these descriptions, a POSITA would have understood that provisioning server 800 is an example of a ***control center computer*** because it performs centralized management and selection functions for the virtualization environment (rather than acting merely as a passive communication node). The provisioning server 800 is ***coupled to*** the host because it communicates with them over network 700 and uses that connection to carry out provisioning and deployment operations directed to those host machines. *Id.*, 9:33-45, 10:27-42.

186. The coupling discussed above is also used for the provisioning

architecture described in Khandekar. MS-1004, 10:27-42, 19:33-45, 21:50-52.

Khandekar describes a centrally managed workflow in which the provisioning server identifies a host, deploys a VM on that host, and invokes host-side interfaces to register the VM. *Id.* This architecture reflects a management relationship in which the control center computer is connected to the computer devices in a manner that allows it to direct provisioning actions within the virtualization environment.

187. A POSITA would have understood that Khandekar's provisioning server 800 is an example of the *control center computer* because, like the "Control Center" computers described in the '050 patent, it is the centralized computer in the virtualization environment that performs the management-side functions recited in claim 9 (e.g., provisioning). The '050 patent specification describes "Control Center computer" (e.g., 110) as having a "Virtual Mapping engine" for "determining the physical resources and virtual assets with which the Control Center will communicate." MS-1001, 7:33-56. Additionally, a "Provisioning Manager" provides "an on-demand solution for provisioning of virtual assets for an automated workflow management," including central management of available and used assets. *Id.*, 8:17-40. Khandekar's provisioning server 800 performs similarly types of functions, since its heuristics engine 880 selects "the most appropriate VM" and also "selects which host machine 1000, 1001 is most appropriate to

provision and deploy the VM on” (MS-1004, 10:27-42) while also coordinates the provisioning workflow and the deployed VMs are “installed for running” on the selected host (*id.*, 10:50-60).

188. Khandekar, the provisioning server 800 and hosts 1000, 1001 operate as a cooperating system connected via network 700, where provisioning operations involve coordinated activity between the two. MS-1004, 21:26-64, 20:54-62. For example, the provisioning engine 810 on the provisioning server calls APIs on the host to register a newly created VM, copies VM configuration and disk files to the host, and issues commands to power on the VM. *Id.*, 21:50-55. The host executes the VM under the VMM and runs configuration scripts that configure the operating system parameters and install applications. *Id.*, 21:55-60. In this way, both the provisioning server 800 and the host contributes processing steps to the provisioning workflow.

189. The cooperative arrangement discussed above in Khandekar’s system is consistent with the architecture described in the ’050 patent specification. Figure 3 of the specification depicts a computer device 300 that includes a CPU 306 and memory 308, alongside a virtualization layer 322 and a Control Agent 324. MS-1001, 9:39-61, FIG. 3. The Control Agent 324 is described as facilitating “communication between the external Control Center components 204-214 (FIG. 2) and the Virtualization Layer 322 of the host machine” and providing “a common

interface to the variety of virtualization software that might be available on the host machine.” MS-1001, 10:18-42. The specification also describes an embodiment in which “the Control Agent 324 can be integrated into the Control Center itself, such that the Control Center can communicate directly with the Virtualization Layer 322 of the host machine.” MS-1001, 10:43-11:4. These disclosures reflect an architecture in which provisioning-related processing is not confined to a single device but is distributed across the control center computer and the computer device in a cooperative arrangement. This is the same type of cooperative architecture that Khandekar describes between provisioning server 800 and hosts 1000, 1001.

[9.2] the virtualization environment having an inventory that includes network resources and virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets;

190. The Khandekar System would have provided a ***virtualization environment*** having an ***inventory*** in the form of database 890 that includes information identifying ***network resources*** and VMs (***virtual assets being software instantiations of computer systems***) hosted on hosts (***computer devices***), with a VMM (***virtualization layer***) executed by the hosts that manage the VMs. MS-1004, 6:1-12, 7:7-27, 14:65-15:14, 18:2-41.

191. As discussed for [1.1] and [1.2], Khandekar describes that the provisioning environment maintains information about hosts and VMs in a

database (e.g., database 890) and uses the information in connection with provisioning operations. MS-1004, 18:2-41. Khandekar also describes that a VM is a “software abstraction” of a physical computer system and that such VMs are deployed on hosts. *Id.*, 6:24-42, 18:2-41. Based on this architecture, Khandekar’s virtualization environment includes an example of an *inventory of the virtual assets* because the system maintains tracked information about the available hosts and VMs as part of the provisioning environment’s managed state. The hosted VMs are examples of *software instantiations of computer systems* because Khandekar describes them as software abstractions of physical computer systems that run on the host. *Id.*, 6:24-42.

192. Khandekar further describes that a VMM is installed on each host and that the VMM “virtualizes all, or at least some of, the resources of the machine.” MS-1004, 7:7-27. The VMM is an example of the *virtualization layer* because it runs on the host and provides the software layer through which the host supports the VMs. The VMM manages the virtual assets because Khandekar describes the VMM as the component that virtualizes host resources for use by the VMs and thereby enables those VMs to operate on the computer device. *Id.*

193. The Khandekar System also includes examples of *network resources* as part of the *virtualization environment*. MS-1004, 6:1-12, 14:65-15:14. Khandekar describes that the host hardware may include “any conventional

network interface device such as one or more cards 150” to enable communication and data transfer with external systems via network 700. *Id.*, 6:1-12. Khandekar also describes that each provisioned VM is provided with one network card in a bridged mode so that the VM becomes available on the LAN.

194. Additionally, the user may request configuration of additional network cards or host-only networking. *Id.*, 14:65-15:14. The physical and virtual network-interface components are examples of ***network resources*** within the virtualization environment because they are network-capable resources used by the hosts and VMs to communicate within and beyond the virtualized system. They are also part of the environment’s inventory because Khandekar describes them as configurable and provisioned resources associated with the managed hosts and VMs. *Id.* For example, the system provides a default network card configuration, allows additional network card configuration, and supports alternative networking modes, which reinforces that these network interface resources are tracked and managed components of the virtualization environment.

[9.3] the control center computer executing a virtual mapping layer that determines an inventory of network resources and virtual assets available to the virtualization environment,

195. The Khandekar System would have provided a provisioning server 800 (***control center computer***) that executes a heuristics engine 880 and maintains a database 890 (collectively ***virtual mapping layer***) that determine, from tracked

information, available hosts, VMs, and network configurations in the virtualization environment (*inventory of network resources and virtual assets available to the virtualization environment*). MS-1004, 10:27-42, 13:1-8, 18:2-41, 20:32-62.

196. As discussed for [1.1] and [1.4], Khandekar describes that database 890 serves as a “central repository of data required by the various components” of the provisioning environment. MS-1004, 18:2-5. Khandekar further describes that the database includes tracked information for hosts, including hardware configuration such as memory size and disk space, as well as tracked information for staged VMs, instantly deployable VMs, and applications. *Id.*, 18:6-41. This database information is an example of an *inventory of virtual assets available to the virtualization environment* because it identifies and maintains the host-side and VM-side information the provisioning server uses to determine what assets are available for provisioning and deployment. The inventory is a type of operational data structure used by the provisioning system to evaluate and select available resources within the virtualization environment. *Id.*, 18:2-41.

197. Khandekar also describes that the heuristics engine 880 “takes this user input and applies heuristics/rules to select the most appropriate VM from a library 830 of staged VMs” and “also selects which host machine 1000, 1001 is most appropriate to provision and deploy the VM on.” MS-1004, 10:27-42.

Khandekar further describes that the heuristics engine scans “the list of available

host machines” and determines the one with enough memory and disk space to host the VM to be created by querying the host through an API. *Id.*, 20:54-58.

Based on these descriptions, a POSITA would have understood from those disclosures that the provisioning server 800 executes management logic that determines which virtual assets and supporting resources are available for use in the virtualization environment. *Id.*, 10:27-42, 20:54-58.

198. Khandekar additionally describes that “[i]nformation about the current state of the different hosts is preferably input dynamically into a database or other data structure in the heuristics engine 880.” MS-1004, 13:1-8. This host-state information is used to suggest components and VM properties that match user requests. *Id.* This dynamic host-state tracking reinforces that the heuristics engine 880 and database 890 collectively determine the currently available inventory of resources in the environment because the system continually updates and consults current resource conditions in deciding which VMs and host resources are available for provisioning. This determination includes network resources because Khandekar describes configurable network-card settings for provisioned VMs and host-side network-interface hardware as part of the environment’s managed resources. *Id.*, 6:1-12, 14:65-15:14. The combined operation of database 890 and heuristics engine 880 therefore represents an example of a ***virtual mapping layer*** executed by the provisioning server 800 (***control center computer***) that determines

the *inventory of network resources and virtual assets available to the virtualization environment.*

[9.4] wherein the virtual mapping layer determines the inventory without regard to processors, operating systems, virtualization platforms, and application software of the virtualization environment; and

199. The Khandekar System would have provided a heuristics engine 880 and database 890 (collectively *virtual mapping layer*) that determine the inventory of available hosts, VMs, and applications in a manner that is independent of specific processors, operating systems, virtualization platforms, and application software. MS-1004, 8:20-35, 11:36-58, 13:17-35, 18:2-41, 20:32-62. Khandekar describes a provisioning environment in which the database catalogs hosts, VMs, and applications across differing configurations and the heuristics engine 880 selects among those available resources based on suitability for the requested deployment. *Id.*

200. As discussed for [9.3], Khandekar describes that database 890 tracks hosts, staged VMs, instantly deployable VMs, and applications across a range of system configurations. MS-1004, 18:2-41. Khandekar also describes that the database catalogs VMs with “different operating systems such as Linux Red Hat, Windows 2000, Windows Me, etc., and/or different versions or releases of a single operating system.” *Id.*, 11:36-49. The applications library also contains code for a wide range of applications, including web servers, database servers, spreadsheets,

word-processing programs, games, and browsers. *Id.*, 11:50-58. A POSITA would have understood from those disclosures that the inventory is determined across multiple operating system and application software variants (rather than being tied to any one particular operating system or software package). This means that the heuristics engine 880 and database 890 (*virtual mapping layer*) determines what is available in the environment independently of (*without regard to*) any single specific operating system or application type. *Id.*, 11:36-58, 18:2-41. The database 890 is also organized to track heterogeneous VM and application options.

201. Khandekar also describes that the heuristics engine 880 selects hosts by scanning “the list of available host machines” and determining the one with enough memory and disk space for the VM to be created. MS-1004, 20:54-58. This host selection logic is driven by resource availability (rather than by commitment to a particular processor architecture, operating-system identity, virtualization platform label, or application-software stack). When a system evaluates available hosts and deployable VMs based on resource sufficiency and requested characteristics, it is determining inventory availability at an abstracted management level rather than by requiring identity-specific matching to a single platform instance. *Id.*, 10:27-42, 20:32-62. This is consistent with the concept of determining inventory without regard to processors, operating systems, virtualization platforms, and application software of the virtualization

environment.

202. Khandekar also describes that “as long as the VMM on each host exports a known hardware interface to the VM deployed on it, then VM deployment will be substantially independent of the actual physical hardware configuration.” MS-1004, 13:17-35. Khandekar further describes that resource mapping may be accomplished transparently to the VM and without modification to either the guest OS or the underlying host operating system, allowing VMs to migrate freely even to other hardware and software platforms. *Id.*, 8:20-35. This reflects an architecture designed to abstract provisioning and deployment away from particular processor and platform details because the VMM exposes a standardized interface to the VM while masking variation in the underlying physical and software environment. In such a system, the management side determination of available inventory operates *without regard to* those underlying processor, operating-system, and virtualization-platform differences in the sense that the system can evaluate and allocate available resources across heterogeneous platforms through the abstraction provided by the VMM. *Id.*, 8:20-35, 13:17-35.

203. A POSITA would have also understood that the a similar type of abstraction extends to application software. MS-1004, 11:50-58, 18:2-41, 20:32-62. Khandekar’s database tracks applications as selectable inventory items and supports deployment of VMs with different software combinations. *Id.*, 11:50-58,

18:2-41. Because the virtual mapping layer determines available inventory across many possible application configurations rather than being limited to one specific application environment, a POSITA would have understood that the inventory determination is made *without regard to* any particular application software of the virtualization environment. In that way, the heuristics engine 880 and database 890 determine the inventory at the virtualization-management level across heterogeneous processors, operating systems, virtualization platforms, and application software.

[9.5] the control center computer executing a provisioning manager that communicates with the virtual mapping layer,

204. The Khandekar System would have provided a provisioning server 800 (*control center computer*) that executes a provisioning engine 810 (*provisioning manager*) that interfaces and exchanges data (*communicates*) with the heuristics engine 880 and database 890 (collectively *virtual mapping layer*).

MS-1004, 10:27-42, 20:32-42, 21:26-34. Khandekar describes that the provisioning engine 810 is a component that ties together the various components of the system and runs each user request as a separate provisioning process. *Id.*, 21:26-31.

205. As discussed for [9.3], the heuristics engine 880 and database 890 together determine the inventory of available hosts, VMs, and applications used in provisioning decisions. MS-1004, 10:27-42, 18:2-41, 20:32-42. Khandekar also

describes that, as part of the provisioning workflow, the provisioning engine 810 “[c]all[s] the heuristics engine 880, to determine the VM’s hardware configuration and the host to deploy on.” *Id.*, 21:33-34. A POSITA would have understood from that disclosure that the provisioning engine 810 communicates with the heuristics engine 880 because the provisioning engine 810 invokes the heuristics engine 880 to obtain the information needed to carry out the provisioning operation. This communication is operational because the provisioning engine 810 depends on the heuristics engine’s output to determine how the requested VM should be configured and where it should be deployed. *Id.*, 10:27-42, 21:26-34.

206. Khandekar further describes that the heuristics engine 880 accepts user input and applies heuristics and rules to select the most appropriate VM and the most appropriate host by matching the user input against information in database 890. MS-1004, 10:27-42, 20:32-42. The provisioning engine 810 therefore also communicates (directly or functionally) with database 890 through a provisioning workflow because the provisioning result returned to the provisioning engine 810 is derived from inventory and state information maintained in the database 890.

207. The provisioning manager 810’s communication with the heuristics engine 880 and database 890 also need not require a separate standalone messaging component. The provisioning engine 810 calls the heuristics engine 880 and relies

on database-backed determinations through ordinary exchange of inputs, queries, and returned provisioning decisions. MS-1004, 10:27-42, 20:32-42. The provisioning engine 810 ties together the system components and runs each provisioning request as a coordinated process reinforces that understanding. *Id.*, 21:26-31.

208. A POSITA would have further understood that this communication is part of a control center architecture because provisioning engine 810, heuristics engine 880, and database 890 are each described as components of the provisioning server environment that cooperate to execute provisioning requests. MS-1004, 10:27-42, 18:2-41, 21:26-34. In this arrangement, the provisioning manager 810 communicates with the heuristics engine 880 and database 890 by obtaining the host selection, VM selection, and configuration information needed to process each request.

209. The provisioning engine 810 also coordinates with the hosts (e.g., 1000, 1001) in executing each provisioning operation. MS-1004, 21:26-64. As described in the provisioning algorithm, after consulting the heuristics engine 880 to determine the appropriate host and VM configuration, the provisioning engine 810 proceeds to copy the VM's configuration and disk files to the host, call the API on the host to register the new VM, connect the VM configuration floppy image to the VM, and call the API on the host to power on the VM. *Id.*, 21:50-55.

Upon powering on, the configuration script in the VM executes on the host to configure operating system parameters and install applications. *Id.*, 21:55-60. If any error occurs during this process, the provisioning engine calls the task manager to abort the task; when the configuration step successfully completes, the provisioning engine calls the API on the host to power the VM down and cleans up the temporary usage area. *Id.*, 21:61-64.

210. The provisioning workflow discussed above is an example where provisioning engine 810 orchestrates a sequence of operations that spans both the provisioning server 800 and the hosts (e.g., 1000, 1001), with each contributing distinct processing steps necessary to complete the provisioning workflow. The provisioning server contributes the steps of consulting the heuristics engine, selecting the host, creating the temporary usage area, and copying files to the host. The host contributes the steps of registering the VM through its API, executing the VM, running the configuration scripts that finalize the operating system and application settings, and returning status information that the provisioning engine uses for error handling and task completion. This is a cooperative processing arrangement in which the provisioning server and the computer collectively perform the provisioning workflow as participants whose processing contributions are coordinated through the provisioning engine's orchestration.

[9.6] the processor being configured to receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request,

211. In the Khandekar System, the provisioning server 800 would have been configured to receive a user-submitted request via a user console (***receive a virtual asset provisioning request***). The request specifies information, such as operating system selection, memory, disk, hostname, and IP configuration used in provisioning the VM to host resources (***parameters to be considered for the virtual asset provisioning request***). MS-1004, 9:46-10:14, 14:31-64, 20:64-67.

212. The “processor” configuration described in [9.6] (and the corresponding operations in [9.6], [9.7], and [9.8]) would have been understood to encompass a complement of processing capabilities distributed across the computer device and the control center computer. This includes a coordinated processor configuration distributed across the computer device and the control center computer. This is because, as I discussed in Section VII.D, the language used in original claim 9 in view of MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) suggests that patentee originally wanted to claim a processor of a Control Center computer (i.e., a centralized server) as perform the operations recited in [9.6], [9.7], and [9.8]. However, patentee’s amendment to claim 9 changed this so that the “processor” is of the “computer device” and not the “control center computer.” To reconcile this

divergence in scope, a POSITA would have looked to the disclosures of MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) and in doing so, would have found teachings rendering obvious a complement of processing capabilities distributed across the computer device and the control center computer.

213. More specifically, MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) describe a host machine's processor as executing complementary local operations (e.g., registering VMs, executing configuration scripts, installing applications) at the Control Center's direction. From these disclosures, a POSITA would have understood that the operative steps recited in [9.6]-[9.8] are drafted with sufficient breadth that they would have been obvious at each of the host and provisioning processors, as those coupled processors operate in coordination with one another.

214. As I explained in Section V.E, I understand that a patent specification must enable a POSITA to make and use the full scope of the claimed invention, and that this enablement requirement applies to the claims as they are written. As I noted above, MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) do not themselves attribute the four provisioning operations to the host machine's processor. Those operations are described only as functions of the Control Center computer. Despite this, the patentee chose to

amend the original claim 9 to attribute those operations to “the processor” of the “computer device” (host machine) and advanced the amended claim on the basis of MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application).

215. From this, patentee’s amendment reflects its view that the disclosures of MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) were sufficient to enable a POSITA to make and use a system in which the host’s processor is configured to perform the four provisioning operations even though MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) attribute those same operations to a central management computer. If that is the case (that is, if disclosure attributing provisioning functions to a central computer is sufficient to teach a POSITA how to implement those functions at a coupled host processor), then Khandekar’s disclosure, which similarly describes a centralized provisioning server performing provisioning operations in coordination with host computers, would likewise be sufficient to teach a POSITA how to make and use the “processor” configuration recited in [9.6]-[9.8]. A POSITA would have therefore found the “processor” configuration recited in [9.6]-[9.8] obvious in view of Khandekar, because Khandekar provides the same type of disclosure that the patentee itself regarded as sufficient to enable the claimed subject matter.

216. For the reasons discussed above, a POSITA reading the claim language of [9.6], [9.7], and [9.8] would understand that, in a coordinated provisioning workflow, the host's processor may satisfy those limitations by performing its own corresponding local provisioning acts within the broader workflow, even where initiation, selection, or higher-level orchestration is performed by the provisioning server. The claim language reasonably encompasses host-side processor participation of the type described in Khandekar's distributed provisioning workflow.

217. A POSITA would have understood that Khandekar describes a provisioning server 800 and host computers (e.g., 1000, 1001) that operate as part of a distributed computing system in which provisioning operations involve coordinated activity between both components. After the provisioning server selects a host and determines the VM configuration, the provisioning server transmits provisioning-related data (e.g., VM configuration files, disk files) to the selected host and invokes host-side APIs to complete deployment operations. MS-1004, 21:50-60. The host is configured to execute complementary operations of those orchestrated by the provisioning server, including registering the VM through a local API, executing configuration scripts that finalize OS parameters, and installing user-selected applications. *Id.* This means that the host's processor performs provisioning-related operations that are initiated by and coordinated

through the provisioning server, rather than operating independently of it.

218. Khandekar's distributed provisioning architecture also reflects a well-known pattern in virtualization environments by the Critical Date. For example, Vasilevsky describes a management architecture in which “commands originate with the management server and are sent to the management agent.” MS-1005, [0088]. These commands include “distributed server operations, instructions to add or remove a node, processor, memory and/or I/O device, instructions to define or delete one or more virtual servers, a node configuration request, virtual server operations, status and logging instructions, heartbeat messages, alert messages, and other miscellaneous operations.” *Id.* In Vasilevsky's architecture, the management server specifies the operations to be performed, and the management agent (executing locally on the host node) implements those operations at the host level. This is the same type of server-initiated, host-executed coordination that Khandekar describes for its provisioning workflows, in which the provisioning server orchestrates deployment and the host's processing resources carry out local complementary steps (e.g., VM registration, configuration script execution, application installation).

219. Vasilevsky's disclosure confirms that a POSITA encountering Khandekar's provisioning architecture would have recognized it as an instance of this well-known pattern (where a centralized server initiates a process and a local

agent on the host implements the process by performing related operations). A POSITA would therefore have understood that Khandekar's host processors (which execute complementary operations coordinated through the provisioning server) are configured to perform provisioning-related operations within the meaning of [9.6]-[9.8]. The allocation of specific processing steps between the server and the host in such architectures was a recognized design variable, not a novel configuration.

220. Khandekar's implementation of provisioning workflow operations on a provisioning server also reflects one of a limited number of recognized architectural choices for where those operations may be executed. By the Critical Date, the allocation of processing responsibilities between a centralized management server and managed host nodes was a well-understood design variable in virtualization environments. For example, Hirschfeld (MS-1006) describes a server cloud management architecture in which the SCM directs PSAs and LSAs executing on physical servers to perform provisioning-related operations locally. During a replication procedure, the PSA on the target physical server "retriev[es] and locally stor[es] a logical server template," "creat[es] logical server configuration information from the logical server template," and "activat[es] a logical server from the logical server configuration information." MS-1006, 4:15-28. During a move procedure, the PSAs on both the original and target physical

servers “manage the changes at the direction of the SCM.” MS-1006, 13:5-23. In each case, the SCM initiates and orchestrates the provisioning operation, and the host-local agents (PSAs, LSAs) execute the corresponding steps on the physical server.

221. Bandhole similarly describes a dynamic computing environment architecture in which a centralized application server 311 orchestrates provisioning and delegates execution-level operations to managers that interact directly with physical machines. The device manager 337 “is configured to receive instructions from device services manager 333” and “operates to install the proper operating system on one or more physical machines when the device service manager 333 requests the particular physical machine with a particular operating system.” MS-1007, [0068]-[0069]. The boot manager (339) similarly “is configured to receive instructions from device service manager 333” and “allocates an IP address for a physical machine before it is started up (i.e., booted up).” MS-1007, [0070].

Hirschfeld and Bandhole each demonstrate the known pattern in which a centralized management server initiates provisioning operations and host-local processing elements execute corresponding steps on the physical machines.

222. A similar pattern is reflected in Khandekar’s provisioning architecture. Khandekar’s provisioning server already distributes selected provisioning steps to the host. For example, after the provisioning server copies

VM configuration and disk files to the host, the host's processing resources register the VM through a local API, execute configuration scripts that finalize OS parameters, and install applications. MS-1004, 21:50-60. These host-side operations (VM registration, OS configuration, application installation) are similar types of operations that Hirschfeld's PSAs and Bandhole's device manager and boot manager perform locally on physical machines at the direction of a centralized server. They each reflect the notion that the boundary between what is executed at the provisioning server versus at the host is a design choice, not a fixed architectural constraint. A POSITA familiar with the architectures described in Hirschfeld and Bandhole would have recognized Khandekar's provisioning server-to-host delegation as an instance of this well-known pattern and would have found it obvious to configure the host's processor to perform provisioning-related operations within the coordinated workflow described in Khandekar.

223. Khandekar's disclosure also makes the operation of *receiv[ing] a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request* obvious. As discussed above, in Khandekar, the provisioning server 800 and hosts 1000, 1001 operate as a cooperating system in which provisioning operations involve coordinated activity between both components. With this architecture, the user console 801 receives provisioning requests from users and transmits them to the provisioning engine 810 on the

provisioning server 800. MS-1004, 13:66-14:3, 15:29-39. Through the user console, the user “specifies the hardware characteristics of the desired machine, such as memory size, number, type and size of disks, number of network cards, etc.” as well as the “desired OS configuration, for example, hostname, IP address.” *Id.*, 9:65-10:14. These user-specified inputs define the configuration characteristics of the VM to be provisioned, including its virtual hardware resources, operating system, and network settings.

224. User-specified inputs are examples of parameters to be considered for a virtual asset provisioning request. This is because each input (e.g., memory size, disk type and size, number of network cards, hostname, IP address) specifies a characteristic that the provisioning system uses to determine how to configure and deploy the requested VM. Khandekar confirms this understanding by describing that the provisioning engine passes the user’s selections to the heuristics engine 880, which then “matches the information against the database 890 to select the appropriate host and the VM hardware configuration.” MS-1004, 20:32-42.

225. The user’s inputs drive the provisioning decisions that follow. Khandekar describes that after the provisioning server selects a host and determines the VM configuration, it transmits provisioning-related information to as selected host (e.g., VM configuration files, disk files) by invoking host-side APIs to complete deployment operations (e.g., VM registration, application

installation). *Id.*, 21:50-60. This represents user-specified inputs reaching the host as part of the provisioning workflow because the host uses the information received from the provisioning server to, for example, register the VM, finalize OS parameters (e.g., hostname, IP address), or install the user-selected applications originated in the user's provisioning request. The receipt of these inputs by the provisioning server is therefore an example of a processor being configured to receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request.

[9.7] automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request,

226. The Khandekar System receives a request with specific parameters (e.g., memory, disk size) and maintains data on the available host inventory. To determine where to deploy the VM, the Khandekar System employs a “heuristics/rules engine 880” which “matches the information [user parameters] against the database 890 to select the appropriate host.” MS-1004, 20:32-42. Specifically, Khandekar explains that the engine operates to “[s]can the list of available host machines 1000, 1001” and “determine the one with enough memory and disk space to host the VM about to be created.” *Id.*, 20:54-58. Once the match is identified, the provisioning engine 810 “deploys a VM on a host” (*id.*, 19:33-34) by copying the necessary files and registering the VM (*id.*, 21:50-52).

227. This automatic provisioning is carried out through coordinated processing between the provisioning server and the host computer. The provisioning server's heuristics engine 880 ensures that the selected host has sufficient resources to satisfy the user-specified parameters, and the provisioning engine 810 configures the VM to match the requested OS and application selections. MS-1004, 14:26-49, 20:43-58. The host computer's processor in turn executes the configuration scripts that finalize the VM's operating system parameters and install the requested applications. MS-1004, 21:55-60. This coordinated workflow is an example of the processor being configured to automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request.

[9.8] assign the provisioned virtual asset to the virtual asset provisioning request and provide user access to the provisioned virtual asset.

228. In the Khandekar System, the complement of processing capabilities distributed across the provisioning server 800 and the host computers (as discussed in my analysis of [9.1] and [9.6]) would have been configured to link a deployed VM to a specific transaction task and temporary usage area generated for the user-submitted request (***assign the provisioned virtual asset to the virtual asset provisioning request***) and enable the user to connect to the deployed VM via remote access software (***provide user access to the provisioned virtual asset***). MS-

1004, 9:65-10:14, 10:61-67, 13:66-14:18, 15:29-39, 19:20-25, 20:64-21:8, 21:39-42, 21:50.

229. As I discussed for limitation [1.6], the provisioning process in the Khandekar System is initiated by a user request, prompting the system to start a new transaction and return a task ID that uniquely “identifies the task.” MS-1004, 15:29-39, 20:64-21:8. To process this request, Khandekar explains that the provisioning server maps the requested VM to this task by “creat[ing] a temporary usage area 860 for the provisioning request that holds the user input as well as the various temporary data needed,” storing the user's configuration choices, and creating the necessary files to copy to the host. *Id.*, 10:61-67, 21:39-42, 21:50.

230. This assignment operation engages both the provisioning server and the host computer. The provisioning server creates the task ID, maintains the temporary usage area, and tracks the association between the request and the resulting VM throughout the provisioning lifecycle. MS-1004, 10:61-67, 19:20-25, 20:64-21:8. The host computer contributes to the assignment by registering the VM through its API and confirming operational status, which is information that the provisioning server uses to determine whether the provisioning task has completed successfully or requires error handling. MS-1004, 21:55-64. This process links the creation and deployment of the specific VM to the task ID generated for the user’s submission. This linkage is maintained through

coordinated processing so that the provisioning server manages the association, and the host computer provides the operational confirmation that enables the provisioning server to close the loop between the provisioned asset and the request.

231. As I discussed for limitation [1.7], once the VM is successfully created and deployed, the user is informed (e.g., “via the browser from which he submitted the request”). MS-1004, 15:29-39. At that point, Khandekar details that “[t]he user can then access the newly created VM via any known remote access software,” such as “software available from VMware.” *Id.*, 14:4-18.

232. Providing user access similarly involves coordinated processing. The provisioning server manages the task lifecycle, informs the user of completion, and provides the interface through which the user initiates a remote connection. The host computer executes the deployed VM and serves as the endpoint to which the user’s remote access software connects. In my opinion, both the provisioning server’s processing (task management, user notification, and connection brokering) and the host’s processing (VM execution and remote session hosting) contribute to providing user access to the provisioned virtual asset. This coordinated workflow is an example of the processor being configured to assign the provisioned virtual asset to the virtual asset provisioning request and provide user access to the provisioned virtual asset.

2. *Claim 11*

[11] The apparatus of claim 9, wherein the provisioning manager provides a user interface for submitting the virtual asset provisioning request.

233. The Khandekar System would have provided a provisioning engine 810 (*provisioning manager*) with a browser-based user console/portal (*user interface*) for submitting provisioning requests. MS-1004, 9:46-64, 13:66-14:3, 14:26-29.

234. As I discussed for limitations [3] and [9.5], in Khandekar, a provisioning server (which executes the provisioning manager) provides a “user console 801” that is “preferably a web server” presenting dialogs for submitting configuration options. MS-1004, 9:46-64, 13:66-14:3, 14:26-29. A POSITA would have understood that, in the Khandekar System, the provisioning manager 810 would have provided a type of browser-based portal/user interface for submitting provisioning requests.

3. *Claim 12*

[12] The apparatus of claim 11, wherein the provisioning manager provides a user interface to access one or more images in a virtual asset image repository.

235. The Khandekar System would have provided a provisioning engine 810 with a user interface that presents available VM images or templates (*one or more images*) stored in a library or database (*virtual asset image repository*) for user selection (*corresponding to access*). MS-1004, 14:4-18, 15:62-67, 17:31-49,

18:14-15.

236. As I discussed for limitation [4], Khandekar explains that the user console 801 presents a list of available staged VMs and instantly deployable VMs, which include image files (e.g., “suspend-to-disk image file”) stored in a repository (e.g., “repository 840” or “library of disk drive images”). MS-1004, 10:15-26, 14:4-18, 15:62-67, 17:31-49, 18:14-15. A POSITA would recognize that the Khandekar System would have thereby provided a user interface that presents available VM images or templates (*one or more images*) stored in a library or database (*virtual asset image repository*) for user selection (*access*).

4. *Claim 14*

[14] The apparatus of claim 9, wherein the inventory further comprises one or more application programs.

237. In the Khandekar System, the management-layer inventory information, such as available VMs and network resources (*inventory*), would have further included data structures for tracking available software applications (*one or more application programs*). MS-1004, 18:24-26, 18:57-19:2.

238. As I discussed for limitations [6] and [9.4], in the Khandekar System, inventory is maintained by the heuristics engine 880 and database 890. This inventory includes data structures for “Applications” that index an “Applications Library 850,” which is “a repository of the software and data needed to install an application 856 in a VM.” MS-1004, 18:24-26, 18:57-19:2. A POSITA would

understand that the inventory and mapping would track this application-availability information alongside the logical/virtual resource pool, such that the tracked inventory data includes one or more application programs available for selection and installation during provisioning.

5. *Claim 16*

[16] The apparatus of claim 14, wherein one of the application programs comprises a network server.

239. In the Khandekar System, tracked data structures for available software applications (*application programs*) would have included code for “web servers” or “database servers” (*network server*). MS-1004, 8:24-26, 18:57-19:2.

240. As I discussed for limitations [8] and [14], the Khandekar System’s inventory (maintained by the heuristics engine 880 and database 890) tracks an Applications Library 850 containing various software applications. MS-1004, 18:24-26, 18:57-19:2. In Khandekar, this library “contain[s] code for any collection of the myriad of existing applications, from web servers, database servers, spreadsheets...” *Id.*, 11:50-58. Khandekar also describes provisioning a network of computers where “a developer wants to test his web application, which uses a web server on one computer, an application server on another...” *Id.*, 23:11-20. By utilizing the heuristics engine 880 and database 890 to track these specific server applications as part of the overall resource pool, a POSITA would have understood that a tracked “web server” or “database server” represents a classic

example of a network server, because they are designed to service requests and provide data or functionality over a network.

B. Ground 1C: Claims 2, 5, 10, And 13 Are Obvious Over The Combination Of Khandekar And Vasilevsky

1. Claim 2

[2] The method of claim 1 further comprising determining one or more performance metrics of the provisioned virtual asset.

241. The Khandekar-Vasilevsky System would have monitored (*determine[ed]*) data such as “performance” and “loading” (corresponding to one or more performance metrics) of the executing VMs (*provisioned virtual asset*) via a management architecture. MS-1004, 13:44-56; MS-1005, [0070], [0072].

242. Khandekar explains that, after provisioning based on a user’s request, “[t]he deployed VMs can be monitored regularly” and that “[h]osts can be monitored for load.” MS-1004, 13:44-56. The Khandekar-Vasilevsky System would also incorporate Vasilevsky’s management-agent architecture as part of the runtime management of those provisioned VMs to enhance Khandekar’s provisioning workflow by providing clearer visibility and runtime control over how these virtual resources consume physical resources. In particular, Vasilevsky describes management communications with an agent executing on the system “for the purpose of performing configuration changes, [and] monitoring performance.” MS-1005, [0070]. Resources are evaluated and adjusted based on the changing requirements of the applications or operating systems, including “performance,

loading, etc.” *Id.*, [0072]. A POSITA would understand that, in this combined system, actively monitoring VM performance and load (or other metrics) provides that the system determines measured runtime information for determining one or more performance metrics associated with the provisioned VMs.

2. *Claim 5*

[5] The method of claim 2, wherein determining the one or more performance metrics further comprises determining one or more real time performance metrics.

243. The Khandekar-Vasilevsky System would have evaluated (*determine[ed]*) the performance-related data (including “performance” and “loading”) for the VMs “dynamically in real time” or “in real-time” (*one or more real time performance metrics*). MS-1005, [0014], [0077].

244. As discussed above for claim 2, the Khandekar-Vasilevsky System would have actively monitored performance and loading conditions for the provisioned VMs. To enable runtime adjustments, Vasilevsky’s system responds to these changing requirements by resizing the virtual server “automatically (e.g., dynamically in real time).” MS-1005, [0014]. The management architecture also enables provisioning of computing resources “in real-time (‘on-the-fly’) without a restart of a virtual computing system.” *Id.*, [0077]. A POSITA would understand that to perform these “on-the-fly” resizing/provisioning actions based on performance and loading conditions, the management system necessarily evaluates

the underlying performance/loading measurements in real time as well.

Accordingly, by monitoring and using performance/loading information to drive “dynamically in real time” adjustments, the Khandekar–Vasilevsky system would be configured to determine one or more real-time performance metrics.

3. *Claim 10*

[10] The apparatus of claim 9, wherein the virtualization environment determines one or more performance metrics of the provisioned virtual asset.

245. In the Khandekar-Vasilevsky System, the provisioning and management architecture (functioning as a virtualization environment) would have monitored data such as “performance” and “loading” (which determines one or more performance metrics) of the executing VMs (representing the provisioned virtual asset). MS-1004, 13:44-56; MS-1005, [0070], [0072].

246. As I discussed for claim 2, in Khandekar, VMs are deployed based on a user’s provisioning request. Once deployed, Khandekar explains that “[t]he deployed VMs can be monitored regularly” and that “[h]osts can be monitored for load.” MS-1004, 13:44-56. The Khandekar-Vasilevsky System would have further incorporated Vasilevsky’s management agent architecture to enhance Khandekar’s provisioning workflow by providing clearer visibility and runtime control over how these virtual resources consume physical resources within the environment. Vasilevsky’s management architecture communicates with a management agent

executing on the system “for the purpose of performing configuration changes, [and] monitoring performance.” MS-1005, [0070]. Resources are evaluated and adjusted based on the changing requirements of the applications or operating systems, including “performance, loading, etc.” *Id.*, [0072]. A POSITA would understand that, in this combined environment, monitoring “performance” and “loading” provides measured runtime information, so that the Khandekar-Vasilevsky System’s virtualization environment would have operated to determine one or more performance metrics of the provisioned virtual asset.

4. *Claim 13*

[13] The apparatus of claim 10, wherein the virtualization environment determines one or more real time performance metrics.

247. In the Khandekar-Vasilevsky System, the virtualization environment would have evaluated (which functions to determine[s]) the performance and loading data of the VMs “dynamically in real time” or “in real-time” (corresponding to one or more real time performance metrics). MS-1005, [0014], [0077].

248. As I discussed for limitations [5] and [10], the virtualization environment of the Khandekar-Vasilevsky System would have actively monitored performance and loading data for the provisioned VMs. To enable runtime adjustments, Vasilevsky’s system responds to these changing requirements by resizing the virtual server “automatically (e.g., dynamically in real time).” MS-

1005, [0014]. The management architecture also enables provisioning of computing resources “in real-time (‘on-the-fly’) without a restart of a virtual computing system.” *Id.*, [0077]. A POSITA would have understood that in order to resize virtual servers dynamically in real time based on changing loading and performance conditions, the virtualization environment must also evaluate those underlying metrics in real time. Thus, by monitoring and utilizing performance data to execute “on-the-fly” resource adjustments, the Khandekar-Vasilevsky System’s virtualization environment would have determined one or more real time performance metrics.

C. Ground 1D: Claims 7 And 15 Are Obvious Over The Combination of Khandekar and Tremain

1. Claim 7

[7] *The method of claim 6, wherein one of the application programs comprises one of a network load balancer, a router, a firewall and a storage server.*

249. In the Khandekar-Tremain System, tracked data structures for available software applications (representing application programs) would have included code for a software-based virtual security firewall (corresponding to a firewall). MS-1004, 11:50-58, 18:24-30, 18:57-19:2; MS-1009, 17:16-19, 32:13-33:10.

250. As I discussed in Ground 1 for claim 6, Khandekar explains that the system maintains application information in the database (e.g., 890), including an

Applications Library 850 that identifies software available for installation. MS-1004, 18:24-30, 18:57-19:2. Khandekar further explains that this library can “contain code for any collection of the myriad of existing applications” that can be selected and installed onto a provisioned VM. *Id.*, 11:50-58. Khandekar also describes that a VM may run an application representing an “database server,” which enables access to a *storage server*. MS-1004, 11:50-58; *see id.*, 23:11-20.

251. A POSITA would have found it obvious to implement Khandekar’s Applications Library to include firewall software in view of Tremain’s teachings regarding software-based virtual firewalls. Doing so would provide an additional security and isolation option for VMs provisioned using Khandekar’s provisioning workflow. Tremain explains that “virtual security firewalls operate similarly to conventional firewalls in that they consist of a special firewall software application which runs on a host operating system on a computer.” MS-1009, 32:13-33:10. Tremain further explains that such a firewall “is preferably implemented by a virtual machine on the real computer... running firewall software.” *Id.*, 17:16-19. A POSITA would therefore understand that, in the combined Khandekar–Tremain system, the tracked application information (i.e., the available application software reflected in the Applications Library) would include code for a software-based virtual security firewall, which functions as a firewall.

2. *Claim 15*

[15] The apparatus of claim 14, wherein one of the application programs comprises one of a network load balancer, a router, a firewall and a storage server.

252. In the Khandekar-Tremain System, the tracked data structures for available software applications (functioning as application programs) would have included code for a software-based virtual security firewall (corresponding to one of a network load balancer, a router, a firewall and a storage server). MS-1004, 11:50-58, 18:24-30, 18:57-19:2; MS-1009, 17:16-19, 32:13-33:10.

253. As I discussed for limitation [14], in the Khandekar-Tremain System, the inventory (maintained by the heuristics engine 880 and database 890) includes an Applications Library 850 containing various software applications. MS-1004, 18:24-30, 18:57-19:2. This library “contain[s] code for any collection of the myriad of existing applications” that can be selected and installed onto a provisioned VM. *Id.*, 11:50-58.

254. A POSITA would have implemented the applications library of the Khandekar System based on the teachings of Tremain to, for instance, include firewall software within the applications library. Such a configuration would have enhanced the security and isolation of VMs provisioned using the system’s automated provisioning techniques. Tremain describes that “virtual security firewalls operate similarly to conventional firewalls in that they consist of a special

firewall software application which runs on a host operating system on a computer.” MS-1009, 32:13-33:10. Such a firewall “is preferably implemented by a virtual machine on the real computer... running firewall software.” *Id.*, 17:16-19. A POSITA would therefore have understood that, in the combined Khandekar-Tremain System, the tracked application information for available software applications (*application programs*) includes code for a software-based virtual security firewall (*firewall*).

D. Ground 2A: Claims 1 And 3 Are Obvious Over Hirschfeld

1. Claim 1

[1.pre] A method for on demand provisioning of virtual asset via a self-service portal in a virtualization system, the method comprising:

255. If the preamble is treated as limiting, Hirschfeld would have rendered the requirements of [1.pre] obvious. Hirschfeld describes a virtualization environment in which the SCM manages a virtualized logical server cloud linked to multiple physical servers, and in which users access logical servers through a web-based portal interface. This corresponds to a virtualization system in which a user may, through a self-service portal, access and initiate operations relating to logical servers that function as the relevant virtual assets. MS-1006, 4:54-5:3, 6:28-63, 6:64-7:30.

256. Hirschfeld describes a “method of virtualizing a plurality of logical servers into a logical server cloud linked to a plurality of physical servers.” MS-

1006, 4:54-5:3. A POSITA would understand this to describe a virtualization system because Hirschfeld's disclosure is directed to creating and managing logical server instances on underlying physical server resources through centralized cloud-management infrastructure. Hirschfeld further explains that the SCM manages the logical server cloud and the physical servers associated with that cloud. *Id.*, 5:66-6:27.

257. Hirschfeld also describes the user-facing portal aspect of that environment. Specifically, Hirschfeld explains that the SCM provides a "web page portal or other access mechanism" through which "a user of any remote terminal" may "login to and access any of the logical servers" of the virtualized logical server cloud via the user interface and the SCM. MS-1006, 6:28-63, 6:64-7:30. The web page portal is an example of the *self-service portal* because it allows the user to access server instances through a browser-accessed interface (rather than requiring the user to manually interact with the underlying physical servers or rely on a separate administrator to expose those functions).

258. A POSITA would have also understood that Hirschfeld's portal is configured as a *self-service portal*. Hirschfeld explains that the user may log in from a remote terminal and access logical servers through the portal interface itself, which indicates that the portal is the user-facing mechanism for obtaining and managing access. MS-1006, 6:28-63, 6:64-7:30. Hirschfeld also emphasizes

that the SCM and user database relieve the user from needing to know the underlying physical-server access information, which is consistent with a self-service abstraction layer that exposes logical-server operations directly to the user. *Id.*, 7:32-62. Hirschfeld also explains that the user interface also provides interfaces enabling SCM control of physical-server and virtualization functions, showing that the portal is not merely informational, but is instead an operational interface through which server-related actions may be invoked. *Id.*, 7:22-30.

259. Hirschfeld also describes the user interface as “provid[ing] interfaces to provide SCM control of the physical server to enable other functions including copy, move, start, shutdown, reset and any other functions enabled by control over the physical server and virtualization software.” MS-1006, 7:22-30. These described functions are the types of operations a POSITA would associate with provisioning or making a virtual asset available within a virtualization system. They show that the portal is used not only for passive access, but also for initiating operational changes affecting logical-server instances under SCM control.

260. Hirschfeld also describes instancing procedures that a POSITA would have understood as examples of provisioning. For example, Hirschfeld explains that a logical server may be replicated or instantiated on a physical server by retrieving a logical-server template, generating resulting configuration information, and activating the logical server from that information. MS-1006, 4:15-28. This

process represents a type of provisioning operation because it involves creating or making available a logical server instance on selected underlying physical server resources through centrally managed control logic. That is, the logical server is not merely being viewed conceptually. It is being instantiated for operation in the managed environment.

261. A POSITA would have also understood that these instancing procedures represent examples of *on-demand provisioning of a virtual asset*. The procedures involve associating a logical server with particular physical server resources through the SCM-controlled management process, which is the functional substance of provisioning in a virtualized environment. MS-1006, 4:15-28, 5:66-6:27. Hirschfeld's portal and control architecture allow a user to interact with logical servers and invoke copy, move, start, and related functions through the user interface, which would indicate to a POSITA that instancing-related operations may be initiated responsively through that same portal framework. MS-1006, 6:64-7:30, 7:22-30. Hirschfeld's emphasis on centralized SCM control and logical server abstraction would have led a POSITA to recognize that the system is designed so that users interact with logical assets on demand through the SCM rather than by manually configuring underlying physical-server details. MS-1006, 2:8-35, 2:50-59, 7:32-62.

262. A POSITA would also have understood the "on demand" aspect from

the way Hirschfeld's portal and SCM architecture are structured. The portal allows a user at a remote terminal to log in and access logical servers through the SCM, while the SCM provides the control layer for server-management and virtualization functions. MS-1006, 6:28-63, 7:22-30. In this type of system, logical server operations can be initiated responsively through the portal as needed by the user, i.e., on demand, rather than only through separate offline administrative intervention.

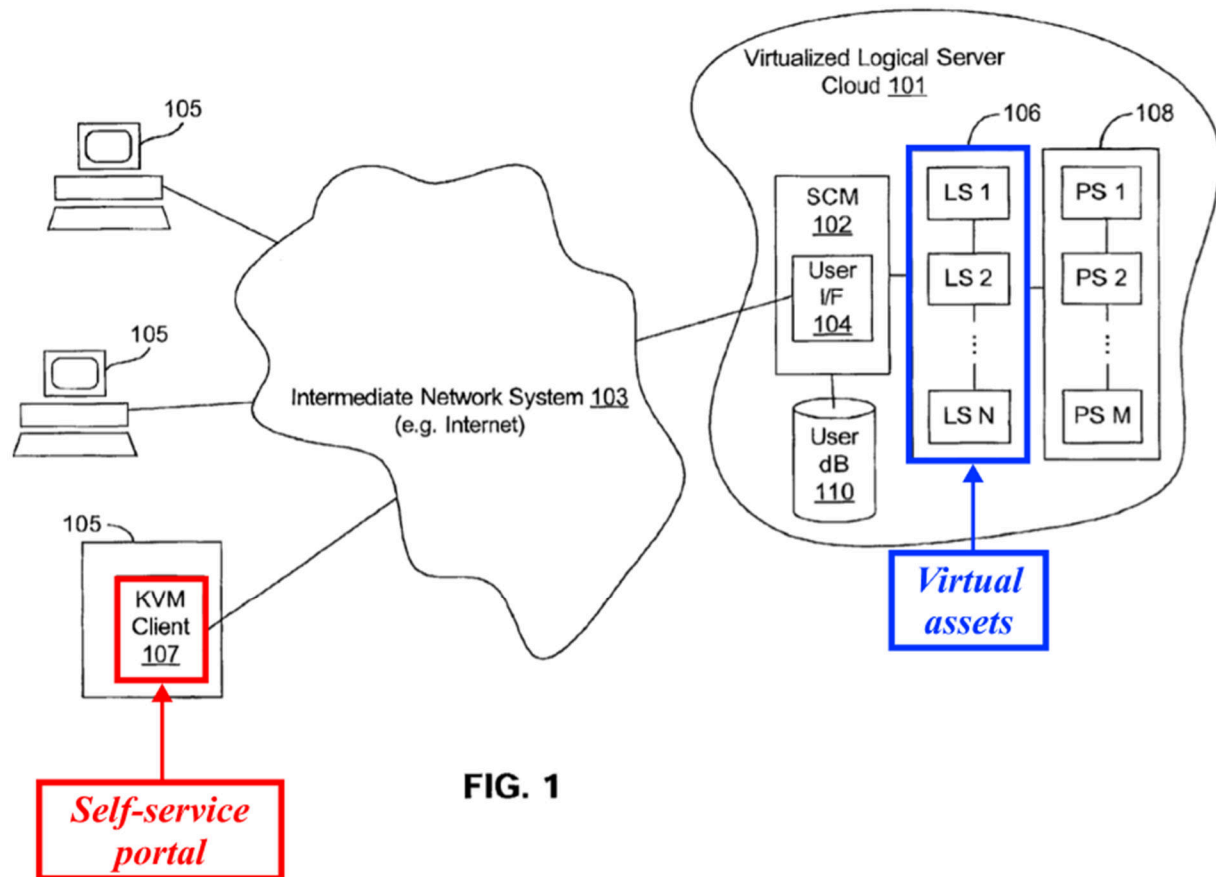


FIG. 1

MS-1006, FIG. 1 (annotated)

[1.1] providing a virtualization system that has an inventory that includes physical resources and virtual assets managed by a virtualization layer running on the physical resources;

263. The Hirschfeld System would have provided a virtualized logical server cloud (***virtualization system***) including physical servers and associated processor, storage, and network resources (***physical resources***), logical servers (***virtual assets***), virtualization software executed on the physical servers that manages the logical servers (***virtualization layer running on the physical resources***), and SCM-maintained status, instance, and tracked information for the logical servers and physical servers (***inventory***). MS-1006, 2:63-3:11, 4:37-52, 8:39-64, 9:35-57, 11:45-57.

264. As I discussed for [1.pre], Hirschfeld describes a “virtualized logical server cloud” that includes logical servers, physical servers, and a server cloud manager. MS-1006, 2:63-3:11. Hirschfeld explains that each physical server is coupled to physical resources including a “network resource,” a “data storage resource” and a “processor resource.” *Id.* A POSITA would have understood these disclosed server-side resources to be examples of the claimed physical resources because they are the underlying hardware-backed resources used to support operation of the virtualized environment and the logical servers instantiated within that environment.

265. Hirschfeld explains that logical servers are created and maintained in

the virtualized logical server cloud and are linked to physical servers while remaining logically distinct from them. MS-1006, 2:63-3:11, 4:37-52. A POSITA would have understood these logical servers to be examples of *virtual assets* because they are server instances provided through virtualization technology rather than standalone physical machines, and because Hirschfeld describes them as existing in a managed logical server cloud over underlying physical-server infrastructure. *Id.*

266. Hirschfeld also describes that each physical server executes virtualization software that virtualizes physical resources for logical servers linked to that physical server. MS-1006, 2:63-3:11; *id.*, 9:35-57. A POSITA would have understood this virtualization software to represent an example of the *virtualization layer running on the physical resources* because Hirschfeld describes it as the software executing on the physical server that enables the physical server's resources to be virtualized and used by the logical servers associated with that server. In this arrangement, the virtualization software is not separate from the physical server. Rather, it is executed on the physical server and manages the virtualized operation of the logical servers supported by that server.

267. The logical servers are managed by that virtualization layer. Hirschfeld describes virtualization software 229 as executing on each physical server and as managing the physical resources of the logical servers linked thereto.

MS-1006, 9:35-57. That disclosure reflects that the virtualization software mediates the relationship between the logical servers and the underlying physical resources on the server.

268. Hirschfeld also describes the SCM as maintaining information regarding the logical servers and their associations with physical servers. Hirschfeld explains that the SCM establishes and maintains “status and instance information” for logical servers, including attributes that link each logical server with a physical server. MS-1006, 2:63-3:20. Hirschfeld also describes a user database that includes “mapping information between users, the logical servers and the physical servers.” *Id.* A POSITA would have understood this maintained status, instance, and mapping information to provide an example of an inventory because it is structured information maintained by the management infrastructure identifying the relevant server instances, the physical resources supporting them, and the relationships among those components.

269. Hirschfeld also describes that tracked information (e.g., resource allocation information) is maintained for logical servers, including non-persistent attributes such as processor resources and memory resources. MS-1006, 12:21-43. Hirschfeld gives examples including “the percentage of CPU time allocated to a particular logical server” and “the amount of memory available to a logical server.” *Id.* Hirschfeld also explains that the SCM may “redistribute processor resources

based on demand and usage.” *Id.* A POSITA would have understood that maintained information about resource allocation and resource availability forms part of the system’s inventory because it reflects what physical resource capacity is assigned, available, or adjustable within the virtualization environment for the benefit of the logical servers.

270. A POSITA would have further understood that this tracked information is an example of inventory of both physical resources and virtual assets available within the virtualization system and their associations. Hirschfeld describes maintained records linking logical servers to physical servers, maintained mapping information between users, logical servers, and physical servers, and maintained resource allocation information describing processor and memory availability. MS-1006, 2:63-3:20, 11:45-57, 12:21-43. These are the types of records a POSITA would have recognized as inventory-like information in a managed virtualization environment because they identify the relevant managed components, their attributes, and their current associations within the system.

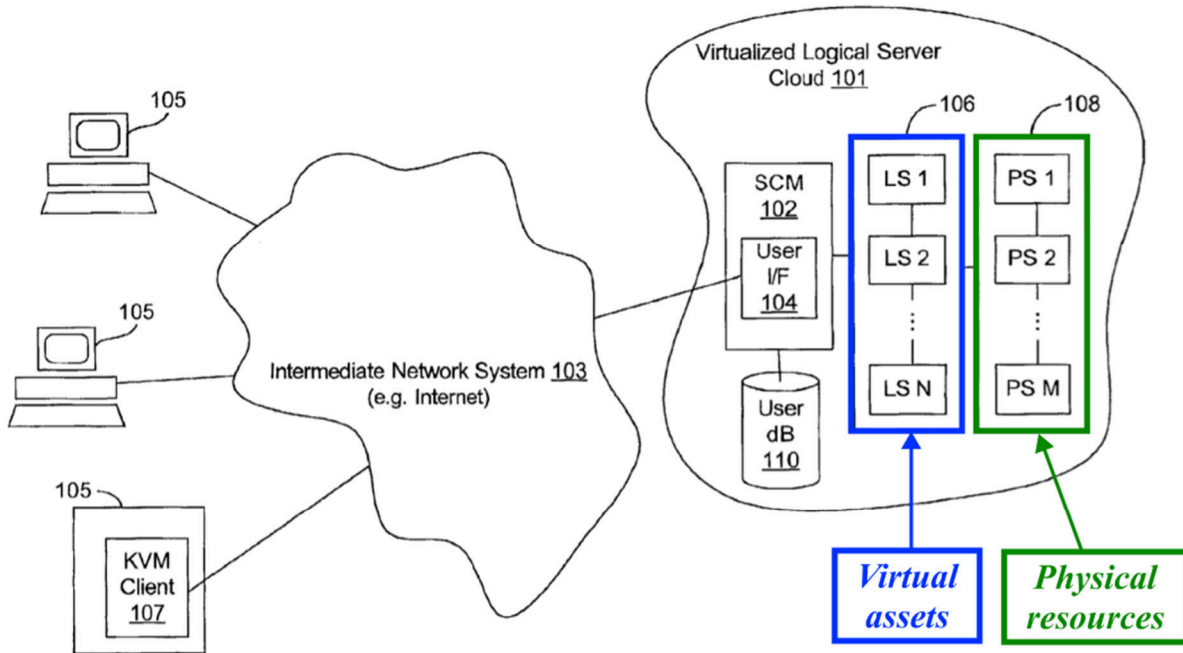


FIG. 1

MS-1006, FIG. 1 (annotated)

[1.2] wherein virtual assets are software instantiations of computer systems;

271. The Hirschfeld System would have provided logical servers that virtually perform the functions of physical devices that connect to computers (*virtual assets*) that include virtualized hardware, operating system, and software layers, and therefore are examples of *software instantiations of computer systems*.

MS-1006, 1:63-2:7, 10:9-27, 10:32-56.

272. As discussed for [1.1], Hirschfeld describes the use of “virtualization technology that enables multiple logical servers to operate on a single physical computer,” including that virtualization allows multiple logical servers to “override

a physical server's attributes and share its resources." MS-1006, 1:63-2:7. A

POSITA would have understood from this disclosure that the logical servers are not separate physical machines, but are instead virtualized server instances implemented through software using the resources of the underlying physical server.

273. Hirschfeld also describes the internal structure of each logical server. Hirschfeld explains that each logical server includes a "hardware" layer, an operating system layer, and a software layer. MS-1006, 10:9-27, 10:32-56. Hirschfeld further describes that the logical server "hardware" layer provides virtualized resources, including virtualized network, storage, memory, and processor resources, corresponding to physical resources of the underlying physical server. *Id.* A POSITA would have understood that this arrangement reflects a computer system instance implemented in software because the logical server presents the functional components of a computer system while relying on virtualization rather than dedicated standalone hardware.

274. A POSITA would have also understood that Hirschfeld's logical servers are examples of *software instantiations of computer systems* because Hirschfeld describes them as server instances that are created through virtualization technology and supported by virtualized hardware resources. MS-1006, 1:63-2:7, 10:9-27, 10:32-56. Hirschfeld's disclosure that each logical server

includes virtualized processor, memory, storage, and network resources would have reinforced that understanding, because those are the types of resources that define the functional architecture of a computer system. *Id.* Hirschfeld's disclosure that multiple such logical servers may operate on a single physical computer would have further indicated to a POSITA that each logical server is a software-implemented instance of a computer system rather than a separate physical machine. MS-1006, 1:63-2:7.

[1.3] receiving a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request;

275. The Hirschfeld System would have received, via the SCM's user interface and web page portal, a user request to associate a logical server (*virtual asset*) with a physical server within the virtualized logical server cloud (*virtual asset provisioning request*). The request would have identified the logical server and thereby specified attributes of that logical server for the SCM to consider in selecting a compatible physical server (*parameters to be considered for the virtual asset provisioning request*). MS-1006, 3:35-44, 6:28-63, 6:64-7:30, 7:22-30, 8:11-38, 11:58-12:19.

276. As discussed for [1.pre], Hirschfeld describes that the SCM provides a "web page portal or other access mechanism" and that a user may use a "local browser to access a web page" and interact with the SCM through the user interface. MS-1006, 6:28-63, 6:64-7:30. Hirschfeld also describes that the user

interface “also provides interfaces to provide SCM control of the physical server to enable other functions including copy, move, start, shutdown, reset and any other functions enabled by control over the physical server and virtualization software.”

Id., 7:22-30. When a user initiates one of these logical server control functions for a selected logical server through the portal, the user is submitting a request that causes the SCM to perform a server side operation affecting the availability or placement of that logical server in the virtualized environment.

277. A POSITA would have also understood that such a request is an example of a provisioning request in this context because Hirschfeld’s SCM-controlled operations include instancing, moving, and associating logical servers with physical servers in the logical server cloud. MS-1006, 4:15-28, 7:22-30, 8:11-38. Hirschfeld describes logical server replication and activation procedures that result in a logical server being instantiated on a physical server using configuration information and template-based control. *Id.*, 4:15-28. Hirschfeld also describes move and other control functions invoked through the portal and mediated by the SCM. *Id.*, 7:22-30. A POSITA would have therefore understood that a user request identifying a logical server for such SCM-controlled action is an example of a ***request to provision that virtual asset within the managed virtualization environment.***

278. Hirschfeld also describes that each logical server has persistent

attributes that “establish its identity,” including “a media access control (MAC) address, a boot disk image, a system identifier, a processor type and access credentials.” MS-1006, 3:35-44. Hirschfeld further describes semi-persistent attributes including “an internet protocol (IP) address, a logical name, a server cloud manager identifier, user account information, a non-boot disk image and network connections.” *Id.*; *see also id.*, 11:58-12:19. These attributes are parameters relevant to provisioning because they define the technical identity, compatibility requirements, and operating characteristics of the logical server that the SCM must preserve or account for when placing or instancing that logical server on physical infrastructure.

279. Hirschfeld also describes that a logical server may be moved to any other “compatible” physical server, where compatibility depends on whether the target server “provides the physical and logical resources needed to retain the persistent attributes of the logical server.” MS-1006, 8:11-38. Hirschfeld explains that this includes physical-resource considerations such as “CPU type, storage capacity, disk configuration, and network connectivity,” as well as logical constraints such as “administrative control” and “software licenses.” *Id.* When a user identifies a particular logical server in a request for instancing, movement, or related SCM-controlled action, the request specifies parameters to be considered because the SCM evaluates the selected logical server’s defined attributes and

constraints in deciding what physical server is compatible for the requested association.

280. A POSITA would have also understood that the parameters need not be re-entered manually in field-by-field form in order to be “specified” by the provisioning request. Hirschfeld describes a system in which the selected logical server carries persistent and semi-persistent attributes maintained by the SCM and used in compatibility determinations. MS-1006, 3:35-44, 8:11-38, 11:58-12:19. In this type of managed virtualization environment, identifying the logical server in the request identifies the attributes associated with that logical server that the SCM uses in carrying out the requested instancing or placement operation.

281. Hirschfeld also describes that reallocation of a logical server may be “directed by the user in the form of an upgrade or downgrade or any other modification requested by the user.” MS-1006, 17:25-39. A POSITA would have understood such user-directed upgrade, downgrade, or modification requests to provide an additional example of a provisioning request that specifies parameters to be considered, because those requests communicate desired changes in the resource profile or operating characteristics of the logical server that the SCM must account for in associating that logical server with appropriate physical resources. See MS-1006, 12:45-13:3, 17:25-39.

[1.4] receiving data on inventory available in the virtualization system;

282. The Hirschfeld System would have received and maintained status, instance, and mapping information for logical servers, physical servers, and user associations in the virtualized logical server cloud (***data on inventory available in the virtualization system***). MS-1006, 2:63-3:20, 3:50-65, 11:4-25, 11:45-57, 12:45-13:3.

283. As discussed for [1.1], Hirschfeld describes that the SCM “establishes and maintains status and instance information” for the logical servers, including “the persistent and any non-persistent attributes that link each logical server with a physical server.” MS-1006, 2:63-3:20. Hirschfeld also describes that a user database is maintained and “includes mapping information between users, the logical servers and the physical servers.” *Id.* A POSITA would have understood that this maintained information represents an example of ***data on the inventory available in the virtualization system*** because it identifies the logical servers present in the system, the physical servers associated with them, and the user-to-server relationships maintained by the SCM.

284. A POSITA would have also understood that the maintained status, instance, and mapping information reflects inventory data because Hirschfeld describes those records as identifying the existence, identity, and associations of the managed server instances in the cloud. MS-1006, 2:63-3:20, 11:45-57. That information indicates which logical servers are present, what attributes they have,

and how they are linked to physical servers and users. *Id.* In a managed virtualization system of this type, those are the kinds of records a POSITA would have recognized as inventory-related information because they describe the components available within the virtualized environment and the relationships among those components.

285. Hirschfeld also describes that the SCM interfaces with and controls physical server agents (“PSAs”) and logical server agents (“LSAs”) to manage physical and logical attributes of logical servers. MS-1006, 3:50-65, 11:4-25, 11:45-57, 12:45-13:3. Hirschfeld explains that a PSA runs on the physical server and is capable of controlling physical-server functions and associated resources through interfaces to the virtualization software. *Id.*, 11:4-25. Hirschfeld further describes that the SCM “manages all of the LSAs” and PSAs, “maintains the status and instance information for each logical server,” and issues instructions to physical servers through corresponding PSAs. *Id.*, 11:45-57, 12:45-13:3.

286. In this architecture, the SCM receives data on inventory available in the virtualization system through its ongoing interaction with the PSAs, LSAs, and underlying servers. Hirschfeld describes an arrangement in which the SCM manages the agents, maintains current status and instance information, and controls logical and physical attributes through those agents. MS-1006, 3:50-65, 11:4-25, 11:45-57, 12:45-13:3. A POSITA would have understood that maintenance of

current status and instance information in such a managed cloud architecture necessarily depends on obtaining information from the managed agents and server-side components regarding the state, attributes, and associations of the logical servers and physical servers. *Id.*

[1.5] automatically provisioning of a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request;

287. The Hirschfeld System would have provisioned, without human intervention (*automatically provisioning*), a particular logical server (*particular virtual asset*) to a selected compatible physical server (*particular physical resource*) under SCM control. The selected physical server satisfies the resource and compatibility requirements associated with the identified logical server (*matches the parameters in the virtual asset provisioning request*). MS-1006, 3:66-4:13, 4:15-28, 8:11-38, 11:45-57, 13:5-23, 15:23-67, 16:1-24.

288. As I discussed for [1.3], Hirschfeld describes that a logical server may be moved to any other “compatible” physical server, and that a compatible physical server is one that “provides the physical and logical resources needed to retain the persistent attributes of the logical server.” MS-1006, 8:11-38. Hirschfeld explains that this includes physical resource considerations, such as “CPU type, storage capacity, disk configuration, and network connectivity” as well as logical constraints, such as “administrative control” and “software licenses.” *Id.* When the

user's request identifies a particular logical server, the SCM evaluates those compatibility requirements in selecting the target physical server for that logical server.

289. A POSITA would have also understood that this selection of a compatible physical server corresponds to provisioning a particular virtual asset to a particular physical resource that matches the parameters in the request. Hirschfeld describes the compatibility requirements by reference to the persistent attributes and associated resource needs of the logical server. MS-1006, 3:35-44, 8:11-38. As explained for [1.3], identifying the logical server in the request identifies the attributes and constraints the SCM must preserve or satisfy in carrying out the requested operation. *Id.* When the SCM selects a compatible physical server that strictly meets those requirements, it is selecting a particular physical resource that matches the parameters implicated by the provisioning request.

290. Hirschfeld also describes automated SCM-controlled procedures that associate logical servers with physical servers. For example, Hirschfeld describes move procedures in which the SCM copies a disk image from a first physical server to a second physical server, copies logical server configuration information, and remaps KVM and network functions to the target physical server. MS-1006, 3:66-4:13, 13:5-23. Hirschfeld also describes replication procedures in which a

physical server agent is instructed to “retrieve[] and locally store[] a logical server template,” “creat[e] logical server configuration information from the logical server template,” and “activat[e] a logical server from the logical server configuration information.” *Id.*, 4:15-28.

291. Hirschfeld’s move and replication procedures are also examples of ***automatic[] provisioning*** operations. Hirschfeld describes the SCM as the control component that maintains status and instance information for each logical server and that issues instructions to physical servers through corresponding PSAs to carry out these operations. MS-1006, 11:45-57. This disclosure reflects that the relevant server association and instancing steps are carried out by the managed system under SCM control, rather than by manual user implementation at the physical-server level. *Id.*

292. These automated procedures also provision a particular logical server to a particular physical server. In the move context, Hirschfeld describes migration of the identified logical server from one physical server to a selected target physical server with corresponding remapping of its operational connections. MS-1006, 3:66-4:13, 13:5-23. In the replication context, Hirschfeld describes activation of a logical server on a physical server based on template-derived configuration information. *Id.*, 4:15-28. In either case, the operation results in a specific logical server being instantiated, activated, or associated on a specific physical server

chosen within the virtualization environment.

293. A POSITA would have further understood that the selected physical server *matches* the parameters in the provisioning request because Hirschfeld ties logical server movement and instancing to compatibility with the logical server's persistent attributes and required resources. MS-1006, 8:11-38. Hirschfeld's description of compatibility in terms of CPU type, storage capacity, disk configuration, network connectivity, administrative control, and software licenses would have indicated to a POSITA that the SCM does not select a target server arbitrarily. *Id.* Rather, the SCM selects a target server whose physical and logical characteristics satisfy the requirements associated with the requested logical server. In that sense, the selected physical resource matches the parameters implicated by the provisioning request.

294. Hirschfeld's description of user-directed upgrade, downgrade, or other modification requests also reinforces this understanding. Hirschfeld explains that reallocation of a logical server may be "directed by the user in the form of an upgrade or downgrade or any other modification requested by the user." MS-1006, 17:25-39. A POSITA would have understood that such user-directed changes require the SCM to select and associate the logical server with physical resources capable of satisfying the modified requirements. *See id.*, 12:45-13:3, 17:25-39.

[1.6] assigning the provisioned virtual asset to the virtual asset provisioning request; and

295. The Hirschfeld System would have assigned the provisioned logical server (*provisioned virtual asset*) to the corresponding user-initiated provisioning request (*virtual asset provisioning request*) by maintaining status, instance, and mapping information for that logical server in SCM-managed data structures, including updated associations to the selected physical server and the requesting user. MS-1006, 2:63-3:20, 7:32-62, 11:45-57, 12:45-13:3, 14:1-17.

296. As discussed for [1.1] and [1.4], Hirschfeld describes that the SCM “establishes and maintains status and instance information for the plurality of logical servers including the persistent and any non-persistent attributes that link each logical server with a physical server.” MS-1006, 2:63-3:11. Hirschfeld also describes that the SCM “maintains the status and instance information for each logical server” on an ongoing basis and “issues instructions to each physical server via corresponding PSAs.” *Id.*, 11:45-57. A POSITA would have understood that this maintained status and instance information reflects the system’s record of which logical server instance has been made available, where it is instantiated, and how it is currently associated within the virtualization environment.

297. Hirschfeld also describes that the SCM tracks logical server configuration information in a local database when a logical server is moved or recreated on a target physical server. MS-1006, 14:1-17. Hirschfeld explains that the database tracking the logical-server configuration information “is reconfigured,

such as modifying links or references, to track the new LS configuration information” on the target physical server. *Id.* This reconfiguration of links or references is an assignment-type operation because it updates the managed records so that the provisioned logical server is tied to the specific resulting server instance and its new physical server association following the requested provisioning event.

298. A POSITA would have also understood that this maintained and updated SCM-side information assigns the provisioned virtual asset to the provisioning request because the provisioning request identifies the logical server to be moved, replicated, or otherwise instantiated. The SCM then updates its managed records to reflect the resulting instantiated or relocated logical-server configuration. MS-1006, 4:15-28, 11:45-57, 14:1-17. In this architecture, the request results in an updated managed server state reflected in the SCM’s maintained instance and configuration records. The system records the outcome of the request by associating the resulting provisioned logical server with the requested action in the SCM-managed data structures.

299. Hirschfeld describes that a user database “incorporates a mapping of users, the logical servers and the physical servers,” including “link information that maps each logical server” with a corresponding physical server. MS-1006, 7:32-48. Hirschfeld also explains that the SCM “automatically accesses the appropriate physical server corresponding to the selected logical server.” *Id.* A POSITA would

have understood that this mapping information provides an additional mechanism by which the system associates the provisioned logical server with the requesting context. This is because the selected logical server remains linked in the database both to the appropriate physical server and to the relevant user access relationship.

300. A POSITA would have also understood that the combination of reconfigured logical server records and maintained user/logical-server mappings satisfies the claimed “assigning” limitation. Hirschfeld describes updated database tracking of the logical server’s new configuration on the target physical server, and it separately describes maintained mapping information between users, logical servers, and physical servers. MS-1006, 7:32-48, 14:1-17. In a managed virtualization system of this type, those maintained records are how the system reflects that a requested provisioning action has resulted in a particular logical server being made available in a particular instantiated form for the relevant user and physical-server context.

301. The SCM’s agent-based control architecture reinforces that understanding. Hirschfeld describes that the SCM manages the relevant status and instance information for each logical server and issues instructions to physical servers through corresponding PSAs in carrying out the provisioning-related operations. MS-1006, 11:45-57, 12:45-13:3. After those operations are carried out, the SCM’s maintained records are the authoritative representation of the resulting

logical-server assignment within the system. That is how the virtualization environment records which logical server has been provisioned, where it resides, and how it is associated to the user and physical-server environment following the request.

[1.7] providing user access to the provisioned virtual asset.

302. The Hirschfeld System would have provided user access to the provisioned logical server (***virtual asset***) through the SCM's user interface and associated access mechanisms, including a web page portal and KVM redirection or mapping, thereby enabling a user of a remote terminal to log in and access the provisioned logical server. MS-1006, 6:28-63, 6:64-7:30, 7:32-62, 14:39-61.

303. As discussed for [1.pre], Hirschfeld describes that the SCM provides a "web page portal or other access mechanism" through which "a user of any remote terminal" may "login to and access any of the logical servers" of the virtualized logical server cloud via the user interface and SCM. MS-1006, 6:28-63. A POSITA would have understood this disclosure to describe provision of user access to the logical server because Hirschfeld identifies the SCM portal as the mechanism through which the user reaches and interacts with the logical-server environment from a remote terminal.

304. Hirschfeld also describes that the user database "incorporates a mapping of users, the logical servers and the physical servers," including "link

information that maps each logical server with one of the physical servers,” such that the SCM “automatically accesses the appropriate physical server corresponding to the selected logical server.” MS-1006, 7:32-62. Hirschfeld further explains that the user “does not have to supply physical server information and, in fact, does not even need to be aware of any specific physical server access or location information.” *Id.* A POSITA would have understood that this mapping-based access architecture provides user access to the provisioned logical server because it allows the system to route the user to the correct server-side instance without requiring the user to know or provide the underlying physical server details.

305. Hirschfeld also describes SCM-managed access mechanisms such as KVM redirection. Hirschfeld explains that the SCM “provides KVM redirection either by providing a KVM interface or by mapping the KVM client to allow direct KVM access to the selected logical server.” MS-1006, 6:64-7:30. Hirschfeld further describes that the SCM may include a KVM service and KVM client that map “KVM functions between the KVM service and the virtual KVM function” of the physical server on which the logical server resides. *Id.*, 14:39-61. These KVM-based functions to be concrete access mechanisms through which the user may interact with the logical server after it has been provisioned to the selected physical server.

306. These KVM redirection and mapping functions provide the operative pathway for access to the provisioned logical server. Hirschfeld describes the SCM as mediating the connection between the user-facing interface and the logical server's server-side execution environment through KVM mapping and related services. MS-1006, 6:64-7:30, 14:39-61. That disclosure shows that access is a technical mechanism by which user interaction is directed to the selected logical server instance residing on the corresponding physical server. *Id.*

307. A POSITA would have further understood that this access is provided to the provisioned logical server. Hirschfeld describes a system in which the SCM maintains mappings between users, logical servers, and physical servers, automatically accesses the appropriate physical server corresponding to the selected logical server, and provides KVM-based access to the selected logical server. MS-1006, 7:32-62, 14:39-61. When read together with Hirschfeld's provisioning, instancing, and movement disclosures, these access mechanisms would have been understood to apply to the resulting provisioned logical-server instance made available through the SCM-managed environment. *See id.*, 4:15-28, 13:5-23.

2. *Claim 3*

[3] *The method of claim 1 further comprising providing a user interface for submitting the virtual asset provisioning request.*

308. The Hirschfeld System would have provided a web page portal and

related SCM user interface (*user interface*) through which a user may initiate instancing and control operations for a logical server (*submitting the virtual asset provisioning request*). MS-1006, 6:28-63, 6:64-7:30, 7:22-30.

309. As discussed for [1.pre] and [1.3], Hirschfeld describes that the SCM provides a “web page portal or other access mechanism” through which “a user of any remote terminal” may “login to and access any of the logical servers” of the virtualized logical server cloud via the user interface and the SCM. MS-1006, 6:28-63, 6:64-7:30. A POSITA would have understood this portal to be a user interface for submitting requests because Hirschfeld describes it as the mechanism through which the user interacts with the SCM to access and control logical servers within the managed virtualization environment.

310. Hirschfeld also describes that the user interface “also provides interfaces to provide SCM control of the physical server to enable other functions including copy, move, start, shutdown, reset and any other functions enabled by control over the physical server and virtualization software.” MS-1006, 7:22-30. A POSITA would have understood that these disclosed interfaces support submission of provisioning-related requests because the identified functions include operations that cause a logical server to be instantiated, moved, started, or otherwise made available under SCM control. *Id.* In a virtualization system of this type, those are the kinds of user-initiated functions through which provisioning requests are

submitted to the management layer.

311. A POSITA would have also understood that, when a user initiates one of these instancing functions for a particular logical server through the portal, the request represents a virtual asset provisioning request. Hirschfeld describes the SCM-controlled system as using logical-server templates, configuration information, and physical-server selection to replicate, activate, and move logical servers. MS-1006, 4:15-28, 7:22-30, 8:11-38. A user's submission through the portal that triggers one of those SCM-controlled operations would therefore have been understood as submission of a request to provision that logical server within the virtualized logical server cloud.

312. The portal is a user interface for submitting the provisioning request because Hirschfeld's architecture places the user-facing interaction point at the SCM while the SCM performs the underlying control and orchestration functions. MS-1006, 6:28-63, 6:64-7:30, 7:22-30. The user interacts with the portal, identifies the logical server, and invokes the relevant control function, while the SCM handles the associated server-side execution. *Id.* That is consistent with a user interface through which the provisioning request is submitted.

E. Ground 2B: Claims 2 And 5 Are Obvious Over The Combination of Hirschfeld And Vasilevsky

1. Claim 2

[2] The method of claim 1 further comprising determining one or more performance metrics of the provisioned virtual asset.

313. The Hirschfeld-Vasilevsky System would have determined data relating to “performance” and “loading” (*one or more performance metrics*) of a provisioned logical server (*provisioned virtual asset*) through the system’s management architecture. MS-1006, 12:21-43; MS-1005, [0070], [0072].

314. Hirschfeld describes a managed logical-server environment in which the SCM maintains information relating to logical-server resource usage and allocation, including processor and memory resource information for particular logical servers. MS-1006, 12:21-43. Hirschfeld describes examples such as “the percentage of CPU time allocated to a particular logical server” and “the amount of memory available to a logical server.” *Id.* A POSITA would have understood that these are the types of system operating characteristics that are monitored and managed in connection with evaluating the behavior and demands of provisioned logical servers.

315. Vasilevsky describes a management architecture in which a manager communicates with management agents “for the purpose of performing configuration changes, monitoring performance, and performing other administrative functions.” MS-1005, [0070]. Vasilevsky also describes that

applications or operating systems may need additional or fewer resources based on “performance” and “loading.” *Id.*, [0072]. A POSITA would have understood these disclosures to describe determination of performance-related measurements for virtualized computing instances because the manager-agent architecture is specifically used to observe and act upon operating conditions of those instances.

316. A POSITA would have understood that incorporating Vasilevsky’s performance-monitoring teachings into Hirschfeld’s managed logical-server environment results in determining one or more performance metrics of the provisioned logical server. Hirschfeld already describes centralized management of logical-server resources and attributes through the SCM. MS-1006, 11:45-57, 12:21-43. Vasilevsky adds a management architecture directed to monitoring “performance” and “loading” of virtualized instances. MS-1005, [0070], [0072]. In that combined system, the monitored performance and loading data would have been understood as performance metrics of the provisioned logical server because they quantify how the provisioned virtualized instance is operating and consuming resources.

317. A POSITA would have also understood that the claimed “determining” requirement is satisfied by active monitoring of performance and loading through the management architecture. Vasilevsky describes manager-agent communications for monitoring performance, and Hirschfeld describes centralized

management of logical-server resource characteristics and allocations. MS-1005, [0070], [0072]; MS-1006, 12:21-43. In a system of this type, obtaining monitored performance and loading information for a provisioned logical server is a determination of performance metrics for that logical server. A POSITA would have therefore understood that the Hirschfeld-Vasilevsky System determines one or more performance metrics of the provisioned virtual asset.

2. *Claim 5*

[5] The method of claim 2, wherein determining the one or more performance metrics further comprises determining one or more real time performance metrics.

318. The Hirschfeld-Vasilevsky System would have determined performance and loading data for logical servers “dynamically in real time” or “in real-time” (*one or more real time performance metrics*) through the system’s management architecture. MS-1005, [0014], [0077].

319. Vasilevsky describes that control of allocating and deallocating resources to a virtual computing system may be provided “dynamically in real time” and describes resource management that occurs “in real-time.” MS-1005, [0014], [0077]. This refers to a management architecture that evaluates the current operational state of virtualized instances while the system is running, rather than only through static or offline assessment. In that context, the monitored “performance” and “loading” data used to drive those runtime decisions are real-

time performance metrics.

320. Vasilevsky describes manager-agent communications for monitoring performance and loading, and it ties that monitoring to dynamic resource-allocation behavior performed in real time. MS-1005, [0070], [0072], [0077]. That relationship indicates that the system is determining current performance-related measurements that are sufficiently timely to support runtime control of the virtualized instance.

321. In the Hirschfeld-Vasilevsky System, those real-time monitoring teachings would have been applied within Hirschfeld's centrally managed logical server environment. Hirschfeld describes an SCM-managed virtualization architecture in which logical servers are maintained and managed through centralized control of logical-server attributes and resource allocations. MS-1006, 11:45-57, 12:21-43. Vasilevsky adds real-time monitoring of performance and loading to that managed environment. MS-1005, [0014], [0070], [0072], [0077]. A POSITA would have understood that, in the combined system, the monitored performance and loading information for provisioned logical servers would therefore represent examples of *one or more real time performance metrics*.

322. A POSITA would have further understood that determining performance data in real time is consistent with Vasilevsky's description of "on-the-fly" or dynamically performed resource adjustments. MS-1005, [0014], [0077].

A system typically does not make resource allocation decisions dynamically in real time based on performance and loading unless it is also determining current performance-related information in real time. The real-time nature of the adjustment logic therefore supports the understanding that the underlying performance and loading inputs are real-time performance metrics.

F. Ground 2C: Claims 6 And 7 Are Obvious Over The Combination of Hirschfeld And Tremain

1. Claim 6

[6] *The method of claim 1, wherein the inventory further comprises one or more application programs.*

323. The Hirschfeld-Tremain System would have provided SCM-maintained status, instance, and mapping information for logical servers and their linkage to physical servers (*inventory*) that further includes information relating to software applications configured on or for those logical servers (*one or more application programs*). MS-1006, 10:9-31, 15:23-67; MS-1009, 17:16-19, 32:13-33:10, 54:4-29, 70:12-28.

324. As discussed for [1.1], Hirschfeld describes that each logical server includes a software layer “for executing desired software applications.” MS-1006, 10:9-31. Hirschfeld also describes a virtualization environment in which the SCM maintains status, instance, and mapping information for logical servers and their associations with physical servers. MS-1006, 2:63-3:20, 11:45-57. Logical servers in the Hirschfeld environment execute software applications as part of their

managed functionality within the logical server cloud.

325. Tremain describes virtual servers in a multi-tenant virtualized hosting environment that are configured with specific application programs to deliver services. MS-1009, 54:4-29. In Tremain, “pre-configured virtual servers” that are “virtual storage container files and include virtual machine operating systems and application and/or server software for the different customers.” *Id.* Tremain also describes virtual servers configured with “operating system and Web server software,” virtual servers “loaded with application software,” and virtual servers “loaded with database management system software.” *Id.*, 70:12-28. A POSITA would have understood these disclosures to show that, in virtualized server environments of this type, application programs are part of the server configuration information relevant to provisioning, instancing, and management.

326. A POSITA would have also understood that, in the Hirschfeld-Tremain System, the inventory maintained by the SCM would further include information about application programs available on or for the logical servers. Hirschfeld already describes SCM-maintained information for logical servers and their associations within the virtualization environment, and Hirschfeld describes logical servers as including a software layer for executing applications. MS-1006, 2:63-3:20, 10:9-31, 11:45-57. Tremain shows that logical or virtual servers in comparable virtualized hosting environments are configured with specific

application and server software as part of the services delivered to users. MS-1009, 54:4-29, 70:12-28. A POSITA would have therefore understood that application-program information would be a natural part of the inventory information maintained for such logical servers.

327. A POSITA would have further understood that application program information would be pertinent to the SCM's management role because Hirschfeld's SCM maintains information about logical-server identity, attributes, and associations, while Tremain shows that the software applications loaded onto the virtual servers are part of the functional identity and service role of those servers. MS-1006, 2:63-3:20, 11:45-57; MS-1009, 17:16-19, 32:13-33:10, 54:4-29, 70:12-28. In a managed virtualization system where logical servers are provisioned and controlled to deliver services, a POSITA would have understood that information about the application programs on those logical servers would be part of the relevant managed inventory.

2. *Claim 7*

[7] The method of claim 6, wherein one of the application programs comprises one of a network load balancer, a router, a firewall and a storage server.

328. The Hirschfeld-Tremain System would have provided inventory information for application programs that includes firewall software implemented for use with a virtual server or logical server (*firewall*). MS-1006, 10:9-31; MS-

1009, 17:16-19, 32:13-33:10.

329. As discussed for [6], in the Hirschfeld-Tremain System, the inventory maintained by the SCM would have included information about application programs available on or for the logical servers. MS-1006, 2:63-3:20, 10:9-31; MS-1009, 54:4-29, 70:12-28. A POSITA would have also understood that one of those application programs could be firewall software based on Tremain's teachings regarding virtual security firewalls in a virtualized hosting environment. MS-1009, 17:16-19, 32:13-33:10.

330. Tremain describes that "virtual security firewalls operate similarly to conventional firewalls in that they consist of a special firewall software application which runs on a host operating system on a computer." MS-1009, 32:13-33:10. Tremain also describes that such a firewall "is preferably implemented by a virtual machine on the real computer" that is "running firewall software." *Id.*, 17:16-19. A POSITA would have understood from these disclosures that Tremain teaches firewall functionality implemented as software for deployment in a virtualized server environment, rather than as a requirement for separate dedicated firewall hardware.

331. A POSITA would have also understood that firewall software of the type described in Tremain would be an example of one of the application programs in the Hirschfeld-Tremain System. Hirschfeld describes logical servers with a

software layer for executing desired software applications. MS-1006, 10:9-31.

Tremain describes firewall software running in a virtualized environment as a software application associated with a virtual machine or virtual server. MS-1009, 17:16-19, 32:13-33:10. In a combined virtualized logical-server environment, a POSITA would have understood that firewall software is a type of application program that could be tracked as part of the application inventory for logical servers.

332. A POSITA would have further understood that including firewall software among the application programs available within Hirschfeld's logical server cloud would have been consistent with Hirschfeld's managed virtualization framework. Hirschfeld describes centralized SCM management of logical servers and their associated information, while Tremain describes virtual security firewalls used to enhance security and isolation in multi-tenant virtualized hosting environments. MS-1006, 2:63-3:20, 11:45-57; MS-1009, 32:13-33:10. A POSITA would have understood that application-program information concerning firewall software would be pertinent to management of logical servers that provide networked services in such an environment.

G. Ground 3: Claims 9-16 Are Obvious Over The Combination Of Khandekar And Bonazzola

1. Overview

333. As I discussed in at least Paragraphs 70-73 and 212-215, MS-1014

(August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) each describe a distributed architecture that renders the “processor” configuration of claim 9 obvious. But, as explained in Section V.F, obviousness is insufficient to establish the patentee’s possession of the claimed subject matter for purposes of priority entitlement.

334. More specifically, counsel has explained to me that in order for a claim of the ’050 patent to get the benefit of a claimed priority date, a POSITA, reading MS-1014 (August 15, 2005 Provisional Application) and MS-1015 (July 1, 2016 Utility Application) must have immediately discerned that the patentee possessed the features described in the claim.

335. Counsel has also explained that the ’050 patent asserts a priority date of August 15, 2005 based on the contents of MS-1014. If claim 9 is not entitled to this date, then a publication of an earlier-filed application to which the ’050 patent claims priority (such as U.S. Patent Application Publication No. 2014/0344462 to Pabari (MS-1012)) is available as prior art.

336. As I discussed in Section VII.D based on the understanding in Section V.F, claim 9 is not entitled to the proclaimed priority date of August 15, 2005 and instead has an actual priority date of October 2, 2017. My analysis below of claim 9 in relation to Pabari-Bonazzola within this section applies this understanding. Specifically, I provide my opinions regarding how the combination of Pabari and

Bonazzola renders claim 9 obvious (both of which are prior art if the actual priority date of claim 9 is October 2, 2017).

2. *Claim 9*

[9.pre] An apparatus that performs automatic management of a virtualization environment in a computer network comprising:

[9.1] a virtualization environment having a computer device with a processor and memory and a control center computer coupled to the computer device,

[9.2] the virtualization environment having an inventory that includes network resources and virtual assets being software instantiations of computer systems hosted on the computer device and a virtualization layer executed by the computer device that manages the virtual assets;

[9.3] the control center computer executing a virtual mapping layer that determines an inventory of network resources and virtual assets available to the virtualization environment,

[9.4] wherein the virtual mapping layer determines the inventory without regard to processors, operating systems, virtualization platforms, and application software of the virtualization environment; and

[9.5] the control center computer executing a provisioning manager that communicates with the virtual mapping layer,

337. The Pabari-Bonazzola System renders limitations [9.pre], [9.1], [9.2], [9.3], [9.4], and [9.5] obvious because Pabari's disclosure, when combined with Bonazzola, satisfies the requirements of the claim elements recited within them. See, e.g., MS-1001, 3:45-65, 6:38-59, 7:33-56, 8:17-40, 9:39-61, FIGS. 1-3.

338. The following chart identifies elements recited in [9.pre], [9.1]-[9.5] and the corresponding disclosure within Pabari that are rendered obvious in combination with Bonazzola. I also cite to corresponding sections of the '050

patent specification for reference.

Limitation	Element	Pabari (MS-1012)
[9.pre]	An apparatus that performs automatic management of a virtualization environment in a computer network	MS-1012, [0018], [0045] MS-1001, 3:45-65, 6:38-59
[9.1]	A computer device with a processor and memory A control center computer coupled to the computer device An inventory that includes network resources and virtual assets being software instantiations of computer systems hosted on the computer device	MS-1001, 6:60-67, 7:1-13, 9:39-67, FIGS. 1-3 MS-1012, [0045], [0046], [0064], FIGS. 1-3
[9.2]	A virtualization layer	MS-1001, 9:62-67, FIG. 3

	executed by the computer device that manages the virtual assets	MS-1012, [0064], FIG. 3
[9.3]	The control center computer executing a virtual mapping layer that determines an inventory of network resources and virtual assets available to the virtualization environment	MS-1001, 7:33-56, FIG. 2 MS-1012, [0049], FIG. 2
[9.4]	The virtual mapping layer determines the inventory without regard to processors, operating systems, virtualization platforms, and application software	MS-1001, 6:38-59; <i>see also</i> 3:45-65 MS-1012, [0045]; <i>see also</i> [0018]
[9.5]	The control center computer executing a	MS-1001, 7:33-56, 8:17-40, FIG. 2 MS-1012, [0049], [0045], FIG. 2

	provisioning manager that communicates with the virtual mapping layer	
--	---	--

[9.6] the processor being configured to receive a virtual asset provisioning request that specifies parameters to be considered for the virtual asset provisioning request,

339. The Pabari-Bonazzola System would have provided a host machine (***computer device***) whose processing device (e.g., 402) (***processor***) executes a setup component (e.g., setup component 130) that receives configuration information (***virtual asset provisioning request***) specifying parameters for provisioning a VM locally on the host (***parameters to be considered for the virtual asset provisioning request***). MS-1013, [0022]-[0023], [0030], FIG. 2 (steps 205-250).

340. Bonazzola describes a setup component 130 that, once installed and running on a host machine, performs a configuration process in which a user is prompted for various pieces of information related to the VM and management system to be installed. MS-1013, [0022]. The configuration information includes storage configuration (e.g., “shared storage connection path and a storage domain and datacenter name”), network configuration (e.g., “Network Interface Card (NIC) information”), VM configuration (e.g., “memory size, number of virtual CPUs for the VM”), and management system configuration (e.g., “host name and

password”). *Id.*, [0023]. The setup component 130 generates installation settings based on the received configuration information. *Id.*, [0030].

341. The configuration information discussed above includes examples of parameters to be considered for a virtual asset provisioning request. This is because each category of configuration information (storage, networking, VM resources, management system settings) represents a characteristic of the VM that will be provisioned on the host. These are the same types of parameters (e.g., memory size, number of CPUs, storage paths, network interfaces) that a POSITA would have recognized as standard inputs for any VM provisioning operation, as they define the virtual hardware, storage, and connectivity characteristics necessary for deploying a functional VM on a host machine. Bonazzola also describes that the setup component uses this configuration information to “generate installation settings” that drive the subsequent provisioning workflow. MS-1013, [0030].

342. Bonazzola also describes a setup component that queries the user for each category of configuration information and generates installation settings based on the received inputs. MS-1013, [0022]-[0023], [0030]. In a system of this type, obtaining user-specified configuration parameters and generating installation settings for provisioning a VM is a receipt and processing of a virtual asset provisioning request that specifies parameters.

[9.7] automatically provision a particular virtual asset to a particular physical resource in the virtualization system that matches the parameters in the virtual asset provisioning request,

343. The Pabari-Bonazzola System would have provided a host machine (computer device) whose processing device (e.g., 402) (processor) executes the setup component 130 to automatically provision a VM (particular virtual asset) to the host machine's physical resources (particular physical resource) matching the user-specified parameters (parameters in the virtual asset provisioning request). MS-1013, [0024], [0032]-[0033], FIG. 3 (steps 305-330).

344. Bonazzola describes that after the configuration process ends, the setup component 130 uses the obtained installation settings to set up network connectivity on the host machine (step 305), prepare a storage location for the VM's virtual disks (step 310), and start the VM using the installation settings, established network connectivity, and prepared storage (step 315). MS-1013, [0032]. The setup component 130 then verifies that the operating system has been installed on the VM (step 325) and restarts the VM upon successful verification (step 330). *Id.*, [0033]. The setup component 130 “performs the noted activities automatically and without user involvement.” *Id.*, [0024].

345. The provisioning workflow discussed above results in a VM whose configuration matches the user-specified parameters. This is because the installation settings used to drive each step of the workflow (network connectivity,

storage preparation, VM creation) are generated from the configuration information provided by the user during the configuration process described in [9.6]. MS-1013, [0023], [0030]. For example, the storage location for virtual disks is prepared based on the user's storage configuration inputs (e.g., shared storage connection path, storage domain), and the VM is started with the number of virtual CPUs and memory size specified by the user during the VM configuration portion of the setup process. *Id.*, [0023], [0032]. There is a correspondence between the user-specified parameters and the resulting VM configuration because the installation settings are derived from those parameters and are applied at each stage of the provisioning workflow.

346. Bonazzola also states that the setup component performs the provisioning activities “automatically and without user involvement.” MS-1013, [0024]. Once the user provides configuration information during the configuration process, the setup component executes the install portion (from network setup through VM creation, OS verification, and restart) without requiring further user interaction. In a system of this type, a host machine's processor executing each provisioning step based on pre-generated installation settings is an example of automatic provisioning of a VM to the host machine's physical resources.

[9.8] assign the provisioned virtual asset to the virtual asset provisioning request and provide user access to the provisioned virtual asset.

347. The Pabari-Bonazzola System would have provided a host machine

(*computer device*) whose processing device (e.g., 402) (*processor*) executes the setup component 130 to assign the provisioned VM (*provisioned virtual asset*) to the provisioning workflow (*virtual asset provisioning request*) and provide user access to the provisioned VM (*provisioned virtual asset*). MS-1013, [0020], [0025]-[0026], FIG. 3 (steps 335-355).

348. As to assigning, Bonazzola describes that after the VM is provisioned and the management system is verified as responsive, the setup component 130 sends a command to the management system to add the host machine to a list of hosts associated with the management system (step 350) and waits for confirmation that the host machine has been added before restarting the VM (step 355). MS-1013, [0026]. In the Pabari-Bonazzola System, the host's processor would have similarly executed a local provisioning component that, upon completing the provisioning of a VM on the host, registers the provisioned VM with the Control Center's management infrastructure (e.g., by communicating with the Control Center to add the host and its provisioned VM to the set of managed resources) and awaits confirmation of successful registration before finalizing the provisioning workflow.

349. The registration and confirmation process discussed above is an example of assigning the provisioned virtual asset to the virtual asset provisioning request. This is because the act of adding the host machine to the management

system's list of managed hosts creates a relationship between the provisioned VM and the management infrastructure that orchestrated its provisioning. Without this registration, the management system would have no awareness of the newly provisioned VM and could not manage it going forward. This linkage (and confirming it before proceeding) is a mechanism by which the system ensures that the provisioned asset is associated with and accountable to the originating provisioning workflow.

350. As to providing user access, Bonazzola describes that the setup component 130 provides “the user a temporary password to connect to the VM 115” during provisioning, enabling direct user access to the provisioned VM on the host machine. MS-1013, [0025]. Once the VM is restarted and the management system is operational, the management system manages “connection protocols between clients 101 and VMs,” which involves ongoing user access to the provisioned VM. *Id.*, [0020]. In the Pabari-Bonazzola System, the host machine's processing device would have similarly enabled user access to the provisioned VM in two phases. The first phase involves providing temporary credentials (e.g., a password) enabling the user to connect directly to the VM during the provisioning process. The second phase involves handing over management of user connectivity to the Control Center's management infrastructure (e.g., through the management system's connection protocol management) once the VM is fully operational and

registered.

351. The two-phase approach discussed above is an example of providing user access to the provisioned virtual asset. During the first phase, the user can connect directly to the VM using the temporary password, which gives the user immediate access to verify or interact with the provisioned VM while it is being finalized. During the second phase, the management system assumes responsibility for managing client-to-VM connections, which is a standard mechanism by which users access VMs in a managed virtualization environment. Temporary direct access during setup is a common practice for initial configuration and verification, and management-system-mediated access is the standard operational model for production VMs.

3. *Claims 10-16*

352. Pabari-Bonazzola System claims 10-16 obvious because Pabari's disclosure, when combined with Bonazzola, satisfies the requirements of the claim elements recited within them.

353. The following chart identifies elements recited in claims 10-16 and the corresponding disclosure within MS-1012 that satisfies the requirements of these limitations. I also cite to the '050 patent specification for reference.

Claim	Element	Pabari (MS-1012) / MS-1001
10	The virtualization	MS-1012, [0061], [0176], FIGS. 2, 18

	environment determines one or more performance metrics of the provisioned virtual asset.	MS-1001, 9:17-27, 22:49-61, FIGS. 2, 18
11	The provisioning manager provides a user interface to access one or more images in a virtual asset image repository.	MS-1012, [0055] MS-1001, 8:17-40
12	The provisioning manager provides a user interface to access one or more images in a virtual asset image repository.	MS-1012, [0055] MS-1001, 8:17-40
13	The virtualization environment determines one or more real time performance metrics.	MS-1012, [0061], [0176], FIG. 18 MS-1001, 9:17-27, 22:49-61, FIG. 18
14	The inventory further comprises one or more	MS-1012, [0048], [0170], FIG. 14 MS-1001, 7:14-32, 21:24-50, FIG. 14

	application programs.	
15	One of the application programs comprises one of a network load balancer, a router, a firewall and a storage server.	MS-1012, [0158], [0168], [0169], FIGS. 12, 13 MS-1001, 18:53-62, 20:38-21:23, 21:51-57, FIGS. 12, 13
16	One of the application programs comprises a network server.	MS-1012, [0166], [0176], FIG. 11 MS-1001, 20:12-32, 22:49-61, FIG. 11

XII. CONCLUSION

354. For all the reasons I have noted in the foregoing paragraphs, claims 1-16 of the '050 Patent are obvious in view of the references discussed above.

355. I currently hold the opinions set expressed in this declaration. But my analysis may continue, and I may acquire additional information and/or attain supplemental insights that may result in added observations.

APPENDIX A

Michael Mitzenmacher
michaelm@eecs.harvard.edu
33 Cary Avenue, Lexington MA 02421
617-496-7172

Research Interests	Design and Analysis of Algorithms; Machine Learning Systems; Networks and Data Transmission; Databases.
Education	UNIVERSITY OF CALIFORNIA AT BERKELEY, Berkeley, CA Ph.D. in Computer Science awarded December, 1996. Dissertation: The Power of Two Choices in Randomized Load Balancing. Advisor: Alistair Sinclair. GPA: 4.0/4.0 CAMBRIDGE UNIVERSITY, Cambridge, England Attended as one of ten recipients of the Churchill Fellowship. Cambridge C.A.S. in Mathematics with highest distinction awarded June 1992. HARVARD COLLEGE, Cambridge, MA B.A. in Mathematics with Computer Science, summa cum laude, awarded June 1991.
Employment	HARVARD UNIVERSITY, Cambridge, MA <i>Spring 1999-present</i> Assistant professor (from Jan. 1999 -July 2002), Associate professor (from July 2002-January 2005), Professor (from Jan. 2005-present), Area Dean for Computer Science (from July 2010-June 2013), Area Co-Chair for Computer Science (from July 2018-June 2019). Teach undergraduate courses including “Introduction to Algorithms and Data Structures” and “Discrete Mathematics for Computer Science”, and graduate courses covering topics in randomized algorithms, algorithms for networks, compression, coding, cryptography, and information retrieval. DIGITAL SYSTEMS RESEARCH CENTER, Palo Alto, CA <i>Fall 1996-Winter 1998</i> Research scientist. Projects included work on information retrieval on the Web, erasure codes, error-correcting codes, on-line algorithms, and load balancing. Co-inventor for twelve submitted patents. SANTA CLARA UNIVERSITY, Santa Clara, CA <i>Spring 1997</i> Guest professor for the undergraduate class “Introduction to Algorithms.” Consultant: I consult on intellectual property issues as an expert witness and in other capacities. As an expert witness, I have testified in multiple trials. I have also consulted for several technology companies and research laboratories, including Adverplex (Cogolabs), Akamai, AT&T, Digital Fountain, eHarmony, Fluent Mobile (Fiksu), Google, Huawei, ITA Software, JobSync, Microsoft, Mitsubishi Research Laboratories, Weta Digital, and Yahoo.
Funding	NSF CCF-2101140: AF: Small: Algorithms and Data Structures with Predictions. PI: Michael Mitzenmacher. Total grant: \$400,000. 7/21-6/24. NSF CNS-2107078: CNS Core: Medium: Approximation and Randomization in the Programmable Data Plane. co-PIs: Minlan Yu. Total grant: \$1,200,000. 7/21-6/25.

NSF DMS-2023528: Foundations of Data Science Institute (Harvard University). 9/20-8/25.

NSF CCF-1563710: CIF: NeTS: Medium: Collaborative Research: Unifying Data Synchronization. co-PIs: David Starobinski, Ari Trachtenberg. Total grant: \$400,000. 7/16-6/20.

NSF CCF-1535705: AitF: FULL: Collaborative Research: Better Hashing for Applications: From Nuts & Bolts to Asymptotics. co-PIs: David Andersen. Total grant: \$250,000. 9/15-8/19.

NSF CCF-1320231: AF: Small: Data Synchronization : Theory, Algorithms, and Practice PI: Michael Mitzenmacher. Total grant: \$399,370. 9/13-8/16.

NSF CNS-1228598: TWC: Medium: Collaborative: Privacy-Preserving Distributed Storage and Computation. co-PIs: Michael Goodrich, Roberto Tamassia. Total grant: \$400,000. 8/12-8/16.

NSF IIS-0964473: HCC: Medium: Collaborative Research: Data-Parallel Hash Tables: Theory, Practice and Applications. co-PIs: Nina Amenta. Total grant: \$171,095. 8/10-7/13.

NSF CCF-0915922: AF : Small : The Theory and Practice of Hash-Based Algorithms and Data Structures. PI: Michael Mitzenmacher. Total grant: \$441,956. 8/09-7/12.

NSF CNS-0721491: NeTS FIND: A Network-Wide Hashing Infrastructure for Monitoring and Measurement. PI: Michael Mitzenmacher. Total grant: \$330,000. 9/07-8/10.

NSF CCF-0634923: Towards a Basic Understanding of Channels with Synchronization Errors. PI: Michael Mitzenmacher. Total grant: \$200,000. 9/06-8/09.

NSF CCR-0121154: ITR/SY Algorithmic Issues in Large Scale Dynamic Networks. co-PIs: Eli Upfal, Brown. Subcontract to Harvard. Total grant: \$502,000. 9/01-8/06.

NSF CCR-0118701: Low Density Parity Check Codes for Channels with Memory. co-PIs: Alek Kavcic. Total grant: \$510,000. 9/01-8/04.

NSF CCR-9983832: Dynamic Processes and Network Algorithms (CAREER). \$200,000. 7/00-6/04.

Dean's Competitive Fund for Promising Scholarship. 2018. \$49,333.

Google University Research Program. 8/13-8/14. PIs: Eddie Kohler and Michael Mitzenmacher \$56,000.

Google University Research Program. 12/11-12/12. \$25,000.

Yahoo! University Research Program. 7/11-6/12. \$10,000.

Google University Research Program. 12/09-12/10. \$60,000.

Yahoo! University Research Program. 9/09-8/10. \$10,000.

Google University Research Program. 8/08-7/09. \$75,000.

Yahoo! University Research Program. 9/07-8/08. \$25,000.

Yahoo! University Research Program. 9/06-8/07. \$50,000.

Cisco University Research Program. 8/08-7/09. \$80,000.

Cisco University Research Program. 8/07-7/08. \$83,000.

Cisco University Research Program. 8/06-7/07. \$80,000.

Cisco University Research Program. 8/05-7/06. \$72,000.

Alfred P. Sloan Research Fellowship. \$40,000. Awarded in 2000.

IBM Faculty Research Grant. \$10,000 Awarded in 2005.

IBM Faculty Research Grant. \$10,000 Awarded in 2003.

Mitsubishi Electronic Research Laboratory. \$10,000 for undergraduate research projects. Awarded in 2002.

Honors

Applied Cryptography and Network Security Conference Test of Time Paper Award (2025)

ACM Paris Kanellakis Theory and Practice Award (2020)

IEEE Fellow (2020)

IEEE Senior Member (2018)

ACM Fellow (2014)

ACM Symposium on Parallelism in Algorithms and Architectures Best Paper Award (2014)

World Wide Web Conference Best Paper Award (2014)

Royal Academy of Engineering Distinguished Visiting Fellowship (2010)

ACM SIGCOMM Test of Time Paper Award (2009)

IEEE Information Theory Society Best Paper Award (2002)

Alfred P. Sloan Research Fellowship (2000)

Sakrisson Award (for Ph.D. thesis at Berkeley) (1997)

NDSEG Graduate Fellowship (1992-95)

NSF Graduate Fellowship Winner (1991)

Churchill Fellowship (1991-92)

Hoopers Prize (for senior thesis at Harvard) (1991)

Phi Beta Kappa (1990)

Harvard Distinction in Teaching Award (1990)

Goldwater Fellowship (1989-91)

Professional Activities

Editorships and Boards:

Advisory Board, Hariri Institute at Boston University (2018-2021)

Editorial Board, Leibniz International Proceedings in Informatics (2009-2019)

Editorial Board, Communications of the ACM (2013-2018)

Science Board, Santa Fe Institute (2013-2019)

Guest Editor, SIAM Journal of Computing special issue for STOC 2009

SIAM Journal on Computing, Editor (2006-2010)

Internet Mathematics, Managing Editor

Guest Editor, Theory of Computing System special issue for SPAA 2002

Journal of Interconnection Networks, Editor

Organizing Committees:

Sublinear Algorithms Workshop Organizing Committee, FODSI (2022)
 Algorithms with Predictions Workshop Organizing Committee, STOC (2022)
 Machine Learning for Algorithms Workshop Organizing Committee, FODSI (2021)
 SIGACT Chair (2015-2018)
 Dagstuhl Seminar: Theory and Applications of Hashing (2017)
 EADS Summer School on Hashing: Theory and Applications (2014)
 ICERM Workshop on Stochastic Graph Models (2014)
 WAW 2013 Organizing Committee
 SIGACT Vice-Chair (2009-2012)
 General Chair, STOC (2010)
 Workshop on Randomized Algorithms and Random Graphs (2010)
 DARPA Information Science and Technology Study Groups (2007-2008)
 SIGACT Committee for the Advancement of Theoretical Computer Science (2005-2008)
 MSRI Workshop on Models of Real-World Random Networks (2005)
 FOCS 2003 Local Arrangements Chair
 Second Workshop on Randomized Algorithms and Random Graphs (2003)
 Workshop on Algorithms and Models for the Web Graph (2002)
 First Workshop on Randomized Algorithms and Random Graphs (2002)
 DIMACS Workshop on Quality of Service Issues in the Internet (2001)
 BU/NSF Workshop on Internet Measurement, Instrumentation, and Characterization (1999)
 FOCS 1998 Local Arrangements Chair

Program Committees:

ACM CoNEXT 2026 Program Committee
 ICLR 2026 Program Committee
 NSDI 2025 Program Committee
 HotNets 2024 Program Committee
 WADS 2023 Program Committee
 ACM CoNEXT 2023 Program Committee
 NeurIPS 2022 Program Committee
 ACM CoNEXT 2022 Program Committee
 ICML 2022 Program Committee
 SIGCOMM 2022 Program Committee
 NeurIPS 2021 Program Committee
 ICML 2021 Program Committee
 ACDA 2021 Program Committee
 ICLR 2021 Program Committee
 APoCS 2021 Program Committee
 ACM CoNEXT 2020 Program Committee
 NeurIPS 2020 Program Committee
 ICML 2020 Program Committee
 APoCS 2020 Program Committee
 NeurIPS 2019 Program Committee
 SIGCOMM 2019 Program Committee
 SOSA 2019 Program Committee (Co-Chair)
 ACM CoNEXT 2018 Program Committee
 ANALCO 2018 Program Committee
 SIGCOMM 2017 Program Committee
 NSDI 2017 Program Committee
 ACM CoNEXT 2016 Program Committee
 SIGCOMM 2016 Program Committee

ALLENEX 2016 Program Committee (Co-Chair)
ICALP 2016 Program Committee (Track C PC Chair)
ACM CoNEXT 2015 Program Committee
ICALP 2015 Program Committee
ITCS 2015 Program Committee
SIGCOMM 2014 Program Committee
ACM CoNEXT 2014 Program Committee
WSDM 2014 Senior Program Committee
WAW 2013 Program Committee (Co-Chair)
STOC 2013 Executive Committee
WSDM 2013 Senior Program Committee
SIGCOMM 2012 Program Committee
ISIT 2012 Program Committee
NSDI 2012 Program Committee
WAW 2011 Program Committee
ACM CoNEXT 2010 Program Committee
SIGCOMM 2010 Program Committee
NSDI 2010 Program Committee
FUN 2010 Program Committee
LATIN 2010 Program Committee
NetSciCom 2009 Program Committee
SIGCOMM 2009 Program Committee
STOC 2009 Program Committee (PC Chair)
NSDI 2009 Program Committee
ICALP 2008 Program Committee
SPAA 2008 Program Committee
WAW 2007 Program Committee
Analysis of Algorithms 2007 Program Committee
ANALCO 2007 Program Committee
ICALP 2007 Program Committee
RANDOM 2007 Program Committee
NCA 2006 Program Committee
STOC 2006 Program Committee
PODC 2005 Program Committee
ANALCO 2005 Program Committee
PODC 2004 Program Committee
SIGCOMM 2004 Program Committee
IPTPS 2004 Program Committee
Data Compression Conference 2004 Program Committee
ESA 2004 Program Committee
WAW 2003 Program Committee
FOCS 2003 Program Committee
Data Compression Conference 2003 Program Committee
ESA 2002 Program Committee
PODC 2002 Program Committee
SPAA 2002 Program Committee
STOC 2002 Program Committee
Data Compression Conference 2002 Program Committee
ALLENEX 2002 Program Committee
HiPC 2001 Program Committee
ISAAC 2000 Program Committee

RANDOM 2000 Program Committee
STACS 2000 Program Committee
FOCS 99 Program Committee
RANDOM 98 Program Committee

Reviewer for several conferences, journals, and grant panels.

Books

Probability and Computing: Randomized Algorithms and Probabilistic Analysis, by Michael Mitzenmacher and Eli Upfal. Published by Cambridge University Press in 2005. (2nd edition available in 2017.) This is a textbook meant for an advanced undergraduate or beginning graduate class. The textbook has been used in courses at Brown, Harvard, U. C. Berkeley, Univ. of Victoria, Tufts, Univ. of Mass. at Amherst, Purdue, U. Penn., and several other universities.

Conference and Journal Publications

“The Mixed Birth-death/death-Birth Moran Process,” with D. Brewster, Y. Huang, and M. Nowak. To appear in *Proceedings of 17th Innovations in Theoretical Computer Science Conference*, 2026.

“Spatiotemporal Sketch Disaggregation: Streaming Analytics with Heterogeneous Resources,” with J. Langlet, P. Chen, Z. Liu, R. Basat, and G. Antichi. To appear in *Proceedings of the 42nd IEEE International Conference on Data Engineering*, 2026.

“Better than Optimal: Improving Adaptive Stochastic Quantization Using Shared Randomness,” with R. Basat, Y. Ben-Itzhak, and S. Vargaftik. To appear in *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS)*, 2026.

“Call on MARS: Scheduling API-Augmented LLM Requests,” with R. Shahout, C. Liang, S. Xin, Q. Lao, Y. Cui, and M. Yu. To appear in *Proceedings of the 39th Conference on Neural Information Processing Systems*, 2025.

“HACK: Homomorphic Acceleration via Compression of the Key-Value Cache for Disaggregated LLM Inference,” with Z. Zhang, H. Shen, S. Vargaftik, R. Basat, and M. Yu. To appear in *Proceedings of ACM SIGCOMM*, 2025.

“Queueing, Predictions, and Large Language Models: Challenges and Open Problems,” with R. Shahout. To appear in *Stochastic Systems*.

“Don’t Stop Me Now: Embedding Based Scheduling For LLMs,” with R. Shahout, E. Malach, C. Liu, W. Jiang, and M. Yu. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

“Parallel Peeling of Invertible Bloom Lookup Tables in a Constant Number of Rounds,” with M. Goodrich and R. Kitagawa. In *Proceedings of the 50th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pp. 70-84, 2025.

“Efficient d -ary Cuckoo Hashing at High Load Factors by Bubbling Up,” with W. Kuszmaul. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pp. 3931-3952, 2025.

“Leveraging Parameterized Chernoff Bounds for Simplified Algorithm Analyses,” with M. Dillencourt and M. Goodrich. *Information Processing Letters*, 187:106516, 2025.

“Optimal and Approximate Adaptive Stochastic Quantization,” with R. Basat, Y. Ben-Itzhak, and S. Vargaftik. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, 2024.

“SkipPredict: When to Invest in Predictions for Scheduling,” with R. Shahout. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, 2024.

“Beyond Throughput and Compression Ratios: Towards High End-to-end Utility of Gradient Compression,” with W. Han, S. Vargaftik, B. Karp, and R. Basat. In *Proceedings of the Twenty-Third ACM Workshop on Hot Topics in Networks*, pp. 186-194, 2024.

“Learning-Augmented Frequency Estimation in Sliding Windows,” with R. Shahout and I. Sabek. In *Proceedings of the Workshop on Intelligent Classification of High-Speed Network Traffic (at ICNP '24)*.

“Learning-Based Heavy Hitters and Flow Frequency Estimation in Streams,” with R. Shahout. In *Proceedings of the 32nd IEEE International Conference on Network Protocols*, 2024.

“Tabula: Efficiently Computing Nonlinear Activation Functions for Secure Neural Network Inference,” with M. Lam, V. J. Reddi, G. Wei, and D. Brooks. In *Transactions on Machine Learning Research*, 2024.

“Accelerating Federated Learning with Quick Distributed Mean Estimation,” with S. Vargaftik, R. Basat, A. Portnoy, G. Mendelson, and Y. Ben-Itzhak. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 3410-3442, 2024.

“THC: Accelerating Distributed Deep Learning Using Tensor Homomorphic Compression,” with M. Li, R. Basat, S. Vargaftik, C. Lao, K. Xu, X. Tang, and, M. Yu. In *Proceedings of NSDI*, pp. 1191-1211, 2024.

“Separating k -Player from t -Player One-Way Communication, with Applications to Data Streams,” with E. Du, D. Woodruff, and G. Yang. In *Theory of Computing*, vol. 19, Article 10, pp. 1-44, 2023.

“Direct Telemetry Access,” with J. Langlet, R. Basat, G. Oliaro, M. Yu, and G. Antichi. In *Proceedings of ACM SIGCOMM*, pp. 832-849, 2023.

“Can Learned Models Replace Hash Functions?” with I. Sabek, K. Vaidya, D. Horn, A. Kipf, and T. Kraska. *Proceedings of the VLDB Endowment*, vol. 16, no. 3, pp. 532-545, 2022.

“Algorithmic Tools for Mining the Motif Structure of Networks,” with B. Matejek, T. Chen, and C. Tsourakakis. To appear in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2022.

“Edge-colored directed subgraph enumeration on the connectome,” with B. Matejek, D. Wei, T. Chen, C. Tsourakakis, and H. Pfister. *Scientific Reports*, 2022.

“Algorithms with Predictions,” with S. Vassilvitskii. *Communications of the ACM*, Volume 65 Issue 7, pp. 33-35, 2022.

“EDEN: Communication-Efficient and Robust Distributed Mean Estimation for Federated Learning,” with S. Vargaftik, R. Basat, A. Portnoy, G. Mendelson, and Y. Ben-Itzhak. In

Proceedings of the 39th International Conference on Machine Learning, pp. 21984-22014, 2022.

“SNARF: A Learning-Enhanced Range Filter,” with K. Vaidya, S. Chatterjee, E. Knorr, S. Idreos, and T. Kraska. In *Proceedings of the VLDB Endowment*, 15(8):1632-1664, 2022.

“Proteus: A Self-Designing Range Filter,” with E. Knorr, B. Lemaire, A. Lim, S. Luo, H. Zhang, and S. Idreos. In *Proceedings of the SIGMOD International Conference on Management of Data*, pp. 1670-1684, 2022.

“Designing Sensitive Viral Diagnostics with Machine Learning,” with H. Metsky, N. Welch, P. Pillai, N. Haradhvala, L. Rumker, S. Mantena, Y. Zhang, D. Yang, C. Ackerman, J. Weller, P. Blainey, C. Myhrvold, and P. Sabeti. *Nature Biotechnology*, 2022.

“The Supermarket Model with Known and Predicted Service Times,” with M. Dell’Amico. *IEEE Transactions on Parallel and Distributed Systems*, 2022.

“Uniform Bounds for Scheduling with Job Size Estimates,” with Z. Scully and I. Grosof. In *Proceedings of 13th Innovations in Theoretical Computer Science Conference*, 114:1-114:30, 2022.

“DRIVE: One-bit Distributed Mean Estimation,” with S. Vargaftik, R. Basat, A. Portnoy, G. Mendelson, and Y. Ben-Itzhak. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, pp. 362-377, 2021.

“Zero-CPU Collection with Direct Telemetry Access,” with J. Langlet, R. Basat, S.S. Ramanathan, G. Oliaro, M. Yu, and G. Antichi. In *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, pp. 108-115, 2021.

“Gradient Disaggregation: Breaking Privacy in Federated Learning by Reconstructing the User Participant Matrix,” with M. Lam, V. J. Reddi, D. Brooks, and G. Wei. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 5959-5968, 2021.

“Putting the ‘Learning’ into Learning-Augmented Algorithms for Frequency Estimation,” with E. Du and F. Wang. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 2860-2869, 2021.

“Queues with Small Advice.” In *Proceedings of the SIAM Conference on Applied and Computational Discrete Algorithms*, pp. 1-12, 2021.

“How to Send a Real Number Using a Single Bit (and Some Shared Randomness),” with R. Basat and S. Vargaftik. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming*, 25:1-25:20, 2021.

“SALSA: Self-Adjusting Lean Streaming Analytics,” with R. Basat, G. Einzinger, and S. Vargaftik. In *The 37th IEEE International Conference on Data Engineering*, pp. 864-875, 2021.

“Partitioned Learned Bloom Filters,” with K. Vaidya, E. Knorr, and T. Kraska. In *The Ninth International Conference on Learning Representations (ICLR)*, 2021.

“Improved Sublinear Time Algorithm for Longest Increasing Subsequence,” with S. Sedighin. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pp. 1934-1947, 2021.

- “DISCOVering the heavy hitters with disaggregated sketches,” with V. Bruschi, R. Basat, Z. Liu, G. Antichi, and G. Bianchi. In *Proceedings of the 16th International Conference on Emerging Networks Experiments and Technologies (CoNEXT)*, pp. 536-537, 2020.
- “Detecting Routing Loops in the Data Plane,” with J. Kucera, R. Basat, M. Kuka, G. Antichi, and M. Yu. In *Proceedings of the 16th International Conference on Emerging Networks Experiments and Technologies (CoNEXT)*, pp. 466-473, 2020.
- “Optimal Learning of Joint Alignments with a Faulty Oracle,” with K. Larsen and C. Tsourakakis. In *Proceedings of ISIT*, pp.2492-2497, 2020.
- “PINT: Probabilistic In-band Network Telemetry,” with R. Basat, S. Ramanathan, Y. Li, G. Antichi, and M. Yu. In *Proceedings of ACM SIGCOMM*, pp. 662-680, 2020.
- “Dynamic Algorithms for LIS and Distance to Monotonicity,” with S. Seddighin. In *Proceedings of the 52nd Annual Symposium on Theory of Computing*, pp. 671-684, 2020.
- “Clustering with a Faulty Oracle,” with K. Larsen and C. Tsourakakis. In *Proceedings of the Web Conference*, pp. 2831-2834, 2020.
- “Prophets, Secretaries, and Maximizing the Probability of Choosing the Best,” with H. Esfandiari, M. Hajiaghayi, and B. Lucier. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, pp. 3717-3727, 2020.
- “Faster and More Accurate Measurement through Additive-Error Counters,” with R. Basat, G. Einziger, and S. Vargaftik. In *Proceedings of IEEE INFOCOM*, pp. 1251-1260, 2020. o “Scheduling with Predictions and the Price of Misprediction.” In *Proceedings of 11th Innovations in Theoretical Computer Science Conference*, 14:1-14:18, 2020.
- “Equitability, Interval Estimation, and Statistical Power,” with Y. Reshef, D. Reshef, and P. Sabeti. *Statistical Science*, (35)2:202-217, 2020.
- “A Digital Fountain Retrospective,” with J. Byers and M. Luby. *ACM SIGCOMM Computer Communication Review*, 49(5), pp. 82-85, 2019.
- “Robust Set Reconciliation via Locality Sensitive Hashing,” with T. Morgan. In *Proceedings of the 38th Symposium on Principles of Database Systems*, pp. 164-181, 2019.
- “Online Pandora’s Boxes and Bandits,” with H. Esfandiari, M. Hajiaghayi, and B. Lucier. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 1885-1892, 2019.
- “Arithmetic Progression Hypergraphs: Examining the Second Moment Method.” In *Proceedings of ANALCO*, pp. 127-134, 2019.
- “A Bayesian Nonparametric View on Count-Min Sketch,” with R. Adams and D. Cai. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 8782-8791, 2018.
- “A Model for Learned Bloom Filters and Optimizing by Sandwiching.” In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 462-471, 2018.
- “Metric Sublinear Algorithms via Linear Sampling,” with H. Esfandiari. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science*, pp. 11-22, 2018.

- “EMOMA: Exact Match in One Memory Access,” with S. Pontarelli and P. Reviriego. *IEEE Transactions on Knowledge and Data Engineering*, 30(11):2120-2133, 2018.
- “Weightless: Lossy Weight Encoding for Deep Neural Network Compression,” with B. Reagen, U. Gupta, R. Adolf, A. Rush, G. Wei, and D. Brooks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4321-4330, 2018.
- “Load Thresholds for Cuckoo Hashing with Double Hashing,” with K. Panagiotou and S. Walzer. In *Proceedings of the 16th Scandinavian Symposium and Workshops on Algorithm Theory*, pp. 29:1-29:9, 2018.
- “Joint Alignment From Pairwise Differences with a Noisy Oracle,” with C. Tsourakakis. In *Proceedings of the 15th Workshop on Algorithms and Models for the Web Graph*, pp. 59-69, 2018. Journal version: *Internet Mathematics*, at <https://doi.org/10.24166/im.06.2019>, December 2020.
- “Reconciling Graphs and Sets of Sets,” with T. Morgan. In *Proceedings of the 37th Symposium on Principles of Database Systems*, pp. 33-47, 2018.
- “Simulated Annealing for JPEG Quantization,” with M. Hopkins and S. Wagner-Carena. In *Proceedings of the 2018 IEEE Data Compression Conference*, p. 414.
- “Adaptive Cuckoo Filters,” with S. Pontarelli and P. Reviriego. In *Proceedings of the ALENEX Conference*, pp. 36-47, 2018. Journal version: *ACM Journal of Experimental Algorithmics*, Article 1.1, March 2020.
- “An Empirical Comparison of the Maximal and Total Information Coefficients to Leading Measures of Dependence,” with D. Reshef, Y. Reshef, and P. Sabeti. *Annals of Applied Statistics*, 12(1):123-155, 2018.
- “Simple Multi-Party Set Reconciliation,” with R. Pagh. *Distributed Computing* 31(6):441-453, 2018.
- “Technical Perspective: Building a Better Hash Function.” *Communications of the ACM*, 60(7), p. 93, 2017.
- “Compresso: Efficient Compression of Segmentation Data For Connectomics,” with D. Haehn, F. Lekschas, B. Matejek, and H. Pfister. In *Proceedings of the 20th International Conference on Medical Image Computing and Computer Assisted Intervention*, pp. 781-788, 2017.
- “Auditable Data Structures,” with M. Goodrich, E. Kornaropoulos, and R. Tamassia. In *Proceedings of the European Symposium on Security and Privacy*, pp. 285-300, 2017.
- “2-3 Cuckoo Filters for Faster Triangle Listing and Set Intersection,” with D. Eppstein, M. Goodrich, and M. Torres. In *Proceedings of the 36th Symposium on Principles of Database Systems*, pp. 247-260, 2017.
- “Scalable motif-aware graph clustering,” with C. Tsourakakis and J. Pachocki. In *Proceedings of the 26th International Conference on the World Wide Web*, pp. 1451-1460, 2017.
- “Measuring Dependence Powerfully and Equitably,” with Y. Reshef, D. Reshef, H. Finucane, and P. Sabeti. *Journal of Machine Learning Research*, 17(212):1-63, 2016.

“Quantized Random Projections and Non-Linear Estimation of Cosine Similarity,” with P. Li and M. Slawski. In *Advances In Neural Information Processing Systems 29*, pp.2748-2756, 2016.

“Analyzing Distributed Join-Idle-Queue: A Fluid Limit Approach.” In *Proceedings of the 54th Annual Allerton Conference on Communication, Control, and Computing*, pp. 312-318, 2016.

“Hardness of Peeling with Stashes,” with V. Nathan. *Information Processing Letters*, 116(11), pp. 682-688, 2016.

“More Practical and Secure History-Independent Hash Tables,” with M. Goodrich, E. Kornaropoulos, and R. Tamassia. In *European Symposium on Research in Computer Security*, pp. 20-38, 2016.

“Models and Algorithms for Graph Watermarking,” with D. Eppstein, M. Goodrich, J. Lam, N. Mamano, and M. Torres. In *Proceedings of the Information Security Conference*, pp. 283-301, 2016.

“Better Bounds for Coalescing-Branching Random Walks on Graphs,” with R. Rajaraman and S. Roche. In *Proceedings of the 28th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 313-323, 2016. Journal version: *ACM Transactions on Parallel Computing*, 5(1), 2:1-2:23, 2018.

“OMASS: One Memory Access Set Separation,” with S. Pontarelli and P. Reviriego. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1940-1943, 2016.

“Voronoi Choice Games”, with M. Boppana, R. Hod, and T. Morgan. In *Proceedings of 43rd International Colloquium on Automata, Languages, and Programming*, 23:1-23:13, 2016.

“Space Lower Bounds for Itemset Frequency Sketches,” with E. Liberty, J. Thaler, and J. Ullman. In *Proceedings of the 35th Symposium on Principles of Database Systems*, pp. 441-454, 2016.

“Technical Perspective: Catching Lies (and Mistakes) in Offloaded Computation,” with J. Thaler. *Communications of the ACM*, 59(2), p. 102, 2016.

“A New Approach to Analyzing Robin Hood Hashing.” In *Proceedings of ANALCO*, pp. 10-24, 2016.

“More Analysis of Double Hashing for Balanced Allocations.” In *Proceedings of ANALCO*, pp. 1-9, 2016.

“Theory Without Experiments: Have We Gone Too Far?” *Communications of the ACM*, 58(9), pp. 40-42, 2015.

“Scaling Up Clustered Network Appliances with XBricks,” with D. Zhou, B. Fan, H. Lim, R. Wang, M. Kaminsky, D. Andersen, and A. Singh. In *Proceedings of ACM SIGCOMM*, 2015.

“Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling,” with J. Pachocki, R. Peng, C. Tsourakakis, and S. Chen Xu. In *Proceedings of the 21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2015.

- “Repeated Deletion Channels,” with B. Haeupler. In *Proceedings of the IEEE Information Theory Workshop*, pp. 152-156, 2014.
- “Multi-Party Set Reconciliation Using Characteristic Polynomials,” with A. Boral. In *Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing*, pp. 1182-1187, 2014.
- “Cleaning Up the Record on the Maximal Information Coefficient and Equitability,” with D. Reshef, Y. Reshef, and P. Sabeti. *Proceedings of the National Academy of Sciences* 111(33):E3362-3, 2014.
- “Cuckoo Filter: Practically Better than Bloom,” with B. Fan, D.G. Andersen, and M. Kaminsky. In *Proceedings of the 10th International Conference on Emerging Networks Experiments and Technologies (CoNEXT)*, pp. 75-88, 2014.
- “Coding for Random Projections,” with P. Li and A. Shrivastava. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pp. 676-684, 2014.
- “Balanced Allocations and Double Hashing.” In *Proceedings of the 26th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 331-342, 2014.
- “Parallel Peeling Algorithms,” with J. Jiang and J. Thaler. In *Proceedings of the 26th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 319-330, 2014. Journal version: *ACM Transactions on Parallel Computing*, 3(1), 7, 2016.
- “Wear Minimization for Cuckoo Hashing: How Not to Throw a Lot of Eggs into One Basket”, with D. Eppstein, M. Goodrich, and P. Pszona. In *Proceedings of the 13th International Symposium on Experimental Algorithms*, pp. 162-173, 2014.
- “Efficient Estimation for High Similarities using Odd Sketches,” with R. Pagh and N. Pham. In *Proceedings of the 23rd International Conference on the World Wide Web*, pp. 109-118, 2014.
- “Improving the Performance of Invertible Bloom Lookup Tables,” with S. Pontarelli and P. Reviriego. *Information Processing Letters*, 114(4), pp. 185-191, 2014.
- “Special Section on the Forty-First Annual ACM Symposium on Theory of Computing (STOC 2009),” with N. Immerlica, J. Katz, R. Servedio, and C. Umans. *SIAM Journal on Computing*, 41(6):1591-1592, 2012.
- “The Daily Deals Marketplace: Empirical Observations and Managerial Implications,” with J. Byers and G. Zervas. *AC SIGecom Exchanges*, 11(2):29-31, 2012.
- “Peeling Arguments and Double Hashing,” with J. Thaler. In *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1118-1125, 2012.
- “The Complexity of Object Reconciliation, and Open Problems Related to Set Difference and Coding,” with G. Varghese. In *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1126-1132, 2012.
- “Cache-Oblivious Dictionaries and Multimaps with Negligible Failure Probability,” with M. Goodrich, D. Hirschberg, and J. Thaler. In *Proceedings of the First Mediterranean Conference on Algorithms*, pp. 203-218, 2012.

- “An Economic Analysis of User-Privacy Options in Ad-Supported Services,” with J. Feigenbaum and G. Zervas. In *Proceedings of the 8th Workshop on Internet and Network Economics*, pp. 30-43, 2012.
- “Verifiable Computation with Massively Parallel Interactive Proofs,” with H. Pfister, M. Roberts, and J. Thaler. In *Proceedings of the USENIX HotCloud Workshop, 2012*.
- “Biff (Bloom Filter) Codes: Fast Error Correction for Large Data Sets,” with G. Varghese. In *Proceedings of the International Symposium on Information Theory*, pp. 483-487, 2012.
- “Continuous Time Channels with Interference,” with I. Ivan, J. Thaler, and H. Yuen. In *Proceedings of the International Symposium on Information Theory*, pp. 860-864, 2012.
- “The Groupon Effect on Yelp Ratings: a Root Cause Analysis,” with J. Byers and G. Zervas. In *Proceedings of the ACM Conference on Electronic Commerce*, pp. 248-265, 2012.
- “Anonymous Card Shuffling and Its Applications to Parallel Mixnets,” with M. Goodrich. In *Proceedings of 39th International Colloquium on Automata, Languages, and Programming*, pp. 549-560, 2012.
- “Chernoff-Hoeffding Bounds for Markov Chains: Generalized and Simplified,” with K. Chung, H. Lam, and Z. Liu. In *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science*, pp. 124-135, 2012.
- “Practical oblivious storage,” with M. Goodrich, O. Ohrimenko and R. Tamassia. In *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*, pp. 13-24, 2012.
- “Hierarchical Heavy Hitters with the Space Saving Algorithm,” T. Steinke and J. Thaler. In *Proceedings of 2012 Meeting on Algorithm Engineering and Experiments*, pp. 160-174.
- “Daily Deals: Prediction, Social Diffusion, and Reputational Ramifications,” with J. Byers and G. Zervas. In *Proceedings of the 5th International Conference on Web Search and Web Data Mining*, pp. 543-552, 2012.
- “Practical Verified Computation with Streaming Interactive Proofs,” with G. Cormode and J. Thaler. In *Proceedings of Innovations in Theoretical Computer Science*, pp. 90-112, 2012.
- “Information Dissemination via Random Walks in d -Dimensional Space,” with H. Lam, Z. Liu, X. Sun, and Y. Wang. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1612-1622, 2012.
- “Privacy-Preserving Group Data Access via Stateless Oblivious RAM Simulation,” with M. Goodrich and R. Tamassia. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 157-167, 2012.
- “An Efficient Rigorous Approach for Identifying Statistically Significant Frequent Itemsets,” with A. Kirsch, A. Pietracaprina, G. Pucci, E. Upfal, and F. Vandin. *Journal of the ACM*, 59(3), 2012.
- “Detecting Novel Associations in Large Datasets,” with D. Reshef, Y. Reshef, H. Finucane, S. Grossman, G. McVean, P. Turnbaugh, E. Lander, and P. Sabeti. *Science*, Vol. 3004, No. 6602, pp. 1518-1524, December 2011.

- “External-Memory Multimaps,” with E. Angelino, M. Goodrich, and J. Thaler. In *Proceedings of the 22nd International Symposium on Algorithms and Computation*, pp. 384-394, 2011. Journal version: *Algorithmica*, 67(1), pp. 23-48, 2013.
- “Brief announcement: large-scale multimaps,” with M. Goodrich. In *Proceedings of the 23rd ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 259-260, 2011.
- “Oblivious RAM Simulation with Efficient Worst-Case Access Overhead,” with M. Goodrich, O. Ohrimenko, and R. Tamassia. In *Proceedings of 3rd ACM Cloud Computing Workshop*, pp. 95-100.
- “Invertible Bloom Lookup Tables,” with M. Goodrich. In *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing*, pp. 792-799, 2011.
- “Cuckoo Hashing with Pages,” with M. Dietzfelbinger and M. Rink. In *Proceedings of the 19th Annual European Symposium on Algorithms*, pp. 615-627, 2011.
- “Network Coding Meets TCP: Theory and Implementation,” with J.K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, and J. Barros. *Proceedings of the IEEE*, 99(3), pp. 490-510, 2011.
- “Privacy-Preserving Access of Outsourced Data via Oblivious RAM Simulation,” with M. Goodrich. In *Proceedings of 38th International Colloquium on Automata, Languages, and Programming*, pp. 576-587, 2011.
- “Graption: A Graph-based P2P Traffic Classification Framework for the Internet Backbone,” with M. Iliofotou, H. Kim, M. Faloutsos, P. Pappu, and G. Varghese. *Computer Networks*, 55(8), pp. 1909-1920, 2011.
- “On the Zero-Error Capacity Threshold for Deletion Channels,” with I. Kash, J. Thaler, J. Ullman. In *Proceedings of the Information Theory and Applications Workshop*, pp. 1-5, 2011.
- “Heapable Sequences and Subsequences,” with J. Byers, B. Heeringa, and G. Zervas. In *Proceedings of ANALCO*, pp. 33-44, 2011.
- “An Introduction to Human Guided Search.” *ACM Crossroads*, 17(2):34-35, 2010.
- “Popularity is Everything: A New Approach to Protecting Passwords from Statistical Guessing Attacks,” with S. Schechter and C. Herley. In *Proceedings of the 5th Usenix Workshop on Hot Topics in Security*, pp. 1-8, 2010.
- “Streaming Graph Computations with a Helpful Advisor,” with G. Cormode and J. Thaler. In *Proceedings of the European Symposium on Algorithms*, pp. 231-242, 2010. Journal version: *Algorithmica*, 65(2), pp. 409-442, 2013.
- “Tight Thresholds for Cuckoo Hashing via XORSAT,” with M. Dietzfelbinger, A. Goerdt, A. Montanari, R. Pagh, and M. Rink. In *Proceedings of 37th International Colloquium on Automata, Languages, and Programming*, pp. 213-225, 2010.
- “Local cluster aggregation models of explosive percolation,” with R.M. D’Souza. *Physical Review Letters*, 14(19), p. 104-107, 2010.

“Tight Asymptotic Bounds for the Deletion Channel with Small Deletion Probabilities,” with A. Kalai and M. Sudan. In *Proceedings of the International Symposium on Information Theory*, pp. 997-1001, 2010.

“An Improved Analysis of the Lossy Difference Aggregator,” with H. Finucane. *Computer Communication Review*, 40(2), pp. 4-11, 2010.

“Information Asymmetries in Pay-Per-Bid Auctions: How Swoopo Makes Bank,” with J. Byers and G. Zervas. In *Proceedings of the ACM Conference on Electronic Commerce*, pp. 1-12, 2010.

“Carousel: Scalable Logging for Intrusion Prevention Systems”, with T. Lam and G. Varghese. In *Proceedings of NSDI 2010*, pp. 361-376.

“AMS Without 4-Wise Independence on Product Domains,” with V. Braverman, K. Chung, Z. Liu, and R. Ostrovsky. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, pp. 119-130, 2010.

“Adaptive Weighing Designs for Keyword Value Computation,” with J. Byers and G. Zervas. In *Proceedings of the Third International Conference on Web Search and Web Data Mining* pp. 331-340, 2010.

“Human-guided search,” with G. Klau, N. Lesh, and J. Marks. *Journal of Heuristics*, 16(3), pp. 289-310, 2010.

“Real-Time Parallel Hashing on the GPU,” with D. Alcantara, A. Sharf, F. Abbasinejad, S. Sengupta, J. Owens, and N. Amenta. *ACM Transactions on Graphics*, 28(5), p. 154, 2009.

“Exploiting Dynamicity in Graph-based Traffic Analysis: Techniques and Applications,” with M. Iliofotou and M. Faloutsos. In *Proceedings of ACM Co-NEXT*, pp. 241-252, 2009.

“An Analysis of Random-Walk Cuckoo Hashing,” with A. Frieze and P. Mellsted. In *Proceedings of 2009 APPROX-RANDOM*, pp. 490-503, 2009. Journal version: *SIAM Journal on Computing*, 40(2), pp. 291-308, 2011.

“Some Open Questions Related to Cuckoo Hashing.” In *Proceedings of the 17th Annual European Symposium on Algorithms*, pp. 1-10, 2009.

“On Compressing Social Networks,” with F. Chierichetti, R. Kumar, S. Lattanzi, A. Panconesi, and P. Raghavan. In *Proceedings of the 15th ACM SIGKDD Int’l Conference on Knowledge Discovery and Data Mining*, pp. 219-228, 2009.

“An Efficient Rigorous Approach for Identifying Statistically Significant Frequent Itemsets,” with A. Kirsch, A. Pietracaprina, G. Pucci, E. Upfal, and F. Vandin. In *Proceedings of the 28th Symposium on Principles of Database Systems*, pp. 117-126, 2009.

“Using the Power of Two Choices to Improve Bloom Filters,” with Steven Lumetta. *Internet Mathematics*, vol. 4. no. 1, pp. 17-33, 2009.

“Network coding meets TCP,” with J. Sundararajan, D. Shah, M. Medard, and J. Barros. In *Proceedings of IEEE INFOCOM*, pp. 280-288, 2009.

- “An Economically Principled Generative Model of AS Graph Connectivity,” with J. Corbo, S. Jain, and D. Parkes. In *Proceedings of IEEE INFOCOM (Mini-conference)*, pp. 2941-2945, 2009.
- “Designing floating codes for expected performance,” with H. Finucane and Z. Liu. In *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*, 2008. Journal version (with F. Chierichetti, H. Finucane, and Z. Liu): *IEEE Transactions on Information Theory*, vol 56, Issue 3, pp. 968-978, 2010.
- “On the performance of multiple choice hash tables with moves on deletions and inserts,” with A. Kirsch. In *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1284-1290, 2008.
- “More Robust Hashing: Cuckoo Hashing with a Stash,” with A. Kirsch and U. Wieder. In *Proceedings of the 16th Annual European Symposium on Algorithms*, pp. 611-622, 2008. Journal version: *SIAM Journal on Computing*, 39(4), pp. 1543-1561, 2009.
- “A survey of results for deletion channels and related synchronization channels.” In *Proceedings of the 2008 Scandinavian Workshop on Algorithm Theory*, pp. 1-3, 2008. Journal version: “New Results and Open Problem for Channels with Synchronization.” *Probability Surveys*, 2009.
- “The Power of One Move: Hashing Schemes for Hardware,” with A. Kirsch. In *Proceedings of IEEE INFOCOM*, pp. 106-110, 2008. Journal version: *IEEE/ACM Transactions on Networking*, 18(6):1752-1765, 2010.
- “Distributed Beamforming with Binary Signaling,” with M. Johnson and K. Ramchandran. In *Proceedings of the 2008 Int’l Symposium on Information Theory (ISIT)*, pp. 890-894, 2008.
- “The Hiring Problem and Lake Wobegon Strategies,” with A. Broder, A. Kirsch, R. Kumar, E. Upfal, and S. Vassilvitskii. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1184-1193, 2008. Journal version: *SIAM Journal on Computing*, 39(4), pp. 1233-1255, 2009.
- “Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream,” with S. Vadhan. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 746-755, 2008. Journal version: with K. Chung and S. Vadhan. *Theory of Computing*, 9, pp. 897-945-1255, 2013.
- “Trace reconstruction with constant deletion probability and related results,” with T. Holenstein, R. Panigrahy, and U. Wieder. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 389-398, 2008.
- “Capacity bounds for sticky channels.” *IEEE Transactions on Information Theory*, vol 54, Issue 1, pp. 72-77, 2008.
- “Network Monitoring Using Traffic Dispersion Graphs,” with M. Iliofotou, P. Pappu, G. Varghese, M. Faloutsos, and S. Singh. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pp. 315-320, 2007.
- “HEXA: Compact Data Structures for Faster Packet Processing,” with S. Kumar, J. Turner, and P. Crowley. In *Proceedings of the IEEE International Conference on Network Protocols*, pp. 246-255, 2007.

- “Using a Queue to De-amortize Cuckoo Hashing in Hardware,” with A. Kirsch and M. Mitzenmacher. In *Proceedings of the 45th Annual Allerton Conference on Communication, Control, and Computing*, 2007.
- “Improved lower bounds for the capacity of i.i.d. deletion and duplication channels, with E. Drinea. *IEEE Transactions on Information Theory*, vol 53, Issue 8, pp. 2693-2714, 2007.
- “Wired Geometric Routing,” with J. Ledlie, P. Pietzuch, and M. Seltzer. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.
- “Capacity Upper Bounds for Deletion Channels,” with S. Diggavi and H. Pfister. In *Proceedings of the 2007 Int’l Symposium on Information Theory (ISIT)*, pp. 1716-1720, 2007.
- “Codes for Deletion and Insertion Channels with Segmented Errors,” with Z. Liu. In *Proceedings of the 2007 Int’l Symposium on Information Theory (ISIT)*, pp. 846-850, 2007. Journal version: *IEEE Transactions on Information Theory*, vol 56, Issue 1, pp. 224-232, 2010.
- “Towards a theory of networked computation : Executive Summary,” with J. Feigenbaum. *ACM SIGACT News*, (4):22-26, 2006.
- “Bloom Filters via d-Left Hashing and Dynamic Bit Reassignment,” with F. Bonomi, R. Panigrahy, S. Singh, and G. Varghese. In *Proceedings of the Allerton Conference*, pp. 877-883, 2006.
- “Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines,” with F. Bonomi, R. Panigrahy, S. Singh, and G. Varghese. In *Proceedings of ACM SIGCOMM*, pp. 315-326, 2006.
- “An Improved Construction for Counting Bloom Filters,” with F. Bonomi, R. Panigrahy, S. Singh, and G. Varghese. In *Proceedings of the European Symposium on Algorithms*, pp. 684-695, 2006.
- “Less Hashing, Same Performance: Building A Better Bloom Filter,” with A. Kirsch. In *Proceedings of the European Symposium on Algorithms*, pp. 456-467, 2006. Journal version: *Random Structures and Algorithms*, 33(2), pp. 187-218, 2008.
- “Stochastic Shortest Paths via Quasi-Convex Maximization,” with M. Brand, J. Kelner, and E. Nikolova. In *Proceedings of the European Symposium on Algorithms*, pp. 552-563, 2006.
- “Polynomial Time Low-Density Parity-Check Codes with Rates Very Close to the Capacity of the q -ary Random Deletion Channel for Large q .” *IEEE Transactions on Information Theory*, vol 52, Issue 12, pp. 5496-5501, 2006.
- “A Simple Lower Bound for the Capacity of the Deletion Channel,” with E. Drinea. *IEEE Transactions on Information Theory*, vol 52, Issue 10, pp. 4657-4660, 2006.
- “On Lower Bounds for the Capacity of Deletion Channels,” with E. Drinea. *IEEE Transactions on Information Theory*, vol 52, Issue 10, pp. 4648-4657, 2006.
- “On the Theory and Practice of Data Recovery with Multiple Versions.” In *Proceedings of the International Symposium on Information Theory*, pp. 982-986, 2006.

- “Fine-Grained Layered Multicast with STAIR,” with J. Byers, Gu-In Kwon, and M. Luby. *IEEE/ACM Transactions on Networking*, 14:1, pp. 81-93, 2006.
- “Distance-Sensitive Bloom Filters,” with A. Kirsch. In *Proceedings of the ALENEX Conference*, pp. 41-50, 2006.
- “BubbleSearch: A Simple Heuristic for Improving Priority-Based Greedy Algorithms,” with N. Lesh. *Information Processing Letters*, 97(4), pp. 161-169, 2006.
- “Network-Aware Overlays with Network Coordinates,” with P. Pietzuch, J. Ledlie, and M. Seltzer. In *Proceedings of the International Workshop on Dynamic Distributed Systems*, 2006.
- “Editorial: The Future of Power Law Research.” *Internet Mathematics*, vol. 2. no. 4, pp. 525-534, 2006.
- “X-Tolerant Test Response Compaction,” with S. Mitra, S. Lumetta, and N. Patil. *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 566-574, Nov/Dec 2005.
- “Privacy preserving keyword searches on remote encrypted data,” with Y.-C. Chang. In *Proceedings of the Third Intl. Conference on Applied Cryptography and Network Security*, pp. 442-455, 2005.
- “Simple Summaries for Multiple Choice Hashing,” with A. Kirsch. In *Proceedings of the 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005. Journal version: *IEEE/ACM Transactions on Networking*: 16(1), pp. 218-231, 2008.
- “A case study in large-scale interactive optimization,” with M. Chimani, N. Lesh, and C. Sidner. In *Proceedings of Artificial Intelligence and Applications 2005*, pp. 24-29.
- “New heuristic and interactive approaches to 2D rectangular strip packing,” with N. Lesh, A. McMahon, and J. Marks. *ACM Journal of Experimental Algorithms*, 10, 2005.
- “Multidimensional Balanced Allocations,” with A. Broder. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 195-196, 2005.
- “Digital Fountains: A Survey and Look Forward.” 2004 Information Theory Workshop.
- “Improved Lower Bounds on the Capacity of I.I.D. Deletion Channels,” with E. Drinea. In *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.
- “Improved Lower Bounds on the Capacity of Deletion Channels,” with E. Drinea. In *Proceedings of the 2004 IEEE International Symposium on Information Theory*.
- “Geometric Generalizations of the Power of Two Choices,” with J. Byers and J. Conside. In *Proceedings of the 16th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 54-63, 2004.
- “Interactive Data Summarization: An Example Application,” with N. Lesh. In *Proceedings of the 2004 Workshop on Advanced Visual Interfaces*, pp. 183-187.
- “On Power Law Distributions for Monkeys Typing Randomly: The Case of Unequal Probabilities,” with B. Conrad. *IEEE Transactions on Information Theory*, 50:7, pp. 1403-1414, 2004.

- “Dynamic Models for File Sizes and Double Pareto Distributions.” *Internet Mathematics*, 1(3):305-334, 2003.
- “A Scaling Result for Explosive Processes,” with R. Oliveira and J. Spencer. *Electronic Journal of Combinatorics*, 11(1), R31, 2004.
- “An Exhaustive Approach to 2D Strip Packing,” with N. Lesh, A. McMahon, and J. Marks. *Information Processing Letters*, 90(1), pp. 7-14, 2004.
- “New Models and Methods for File Size Distributions,” with B. Tworetzky. In *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, pp. 603-612, 2003.
- “Capacity Approaching Signal Constellations for Channels with Memory,” with A. Kavčić, and X. Ma. *IEEE Transactions on Information Theory*, 49:7, pp. 1636-1652, 2003.
- “The Power Spectra of Good Codes for Partial Response Channels,” with A. Kavcic and X. Ma. In *Proceedings of the 2003 IEEE International Symposium on Information Theory*, p. 45.
- “Concatenated Codes for Deletion Channels,” with J. Chen, C. Ng, and N. Varnica. In *Proceedings of the 2003 IEEE International Symposium on Information Theory*, p. 218.
- “Verification Codes for Deletion Channels.” In *Proceedings of the 2003 IEEE International Symposium on Information Theory*, p. 217.
- “Estimating and Comparing Entropies Across Written Natural Languages Using PPM Compression,” with F. Behr, V. Fossum, and D. Xiao. In *Proceedings of the 2003 IEEE Data Compression Conference*, p. 416.
- “Simple Load Balancing for Distributed Hash Tables,” with J. Byers, and J. Considine. Appears in the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), pp. 80-87, 2003.
- “A Complete and Effective Move Set for Simplified Protein Folding,” with N. Lesh, and S. Whitesides. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology*, pp 188-195, 2003.
- “Network Applications of Bloom Filters: A Survey,” with A. Broder. In *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, pp. 636-646, 2002. Journal version: *Internet Mathematics*, 1(4):485-509, 2003.
- “Verification Codes,” with M. Luby. In *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, pp. 38-47, 2002. Journal version: *IEEE Transactions on Information Theory*: 51(1), pp. 120-127, 2005.
- “Balls and Bins with Memory,” with B. Prabhakar and D. Shah. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 799-808, 2002.
- “Informed Content Delivery Across Adaptive Overlay Networks,” with J. Byers, J. Considine, and S. Rost. In *Proceedings of ACM SIGCOMM '02*, available as *Computer Communications Review*, vol. 32:4, pp. 47-60, 2002. Journal version: *IEEE/ACM Transactions on Networking*, 12:5, pp. 767-780, 2004.

- “Optimal Plans for Aggregation,” with A. Broder. In *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing*, pp. 144-152, 2002.
- “Human-Guided Tabu Search,” with G. Klau, N. Lesh, and J. Marks. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 41-47, 2002.
- “The HuGS Platform: A Toolkit for Interactive Optimization,” with G. Klau, N. Lesh, J. Marks, and G. Schafer. In *Proceedings of the Workshop on Advanced Visual Interfaces*, pp. 324-330, 2002.
- “Exact Sampling of TCP Window States,” with A. Goel. In *Proceedings of IEEE INFOCOM*, pp. 259-265, 2002.
- “Balls and Bins Models with Feedback,” with E. Drinea and A. Frieze. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 308-315, 2002.
- “A Brief History of Lognormal and Power Law Distributions.” In *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, pp. 182-191, 2001. Journal version: *Internet Mathematics*, 1(2):226-251, 2003.
- “Capacity Approaching Signal Constellations for Channels with Memory,” with A. Kavčić, X. Ma, and N. Varnica. In *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, pp. 311-320, 2001.
- “Compressed Bloom Filters.” In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing*, pp. 144-150, 2001. Journal version: *IEEE/ACM Transactions on Networks*, 10:5, pp. 613-620, October 2002.
- “Towards More Completed Models of TCP Throughput,” with R. Rajaraman. *Journal of Supercomputing*, 20(2), pp. 137-160, September 2001.
- “Deriving Performance Bounds for ISI Channels Using Gallager Codes,” with A. Kavčić, B. Marcus, and B. Wilson. In *Proceedings of the 2001 International Symposium on Information Theory*, p. 345, 2001.
- “Toward Compressing Web Graphs,” with M. Adler. In *Proceedings of the 2001 Data Compression Conference*, pp. 203-212, 2001.
- “On the Hardness of Finding Multiple Preset Dictionaries.” In *Proceedings of the 2001 Data Compression Conference*, pp. 411-418, 2001. Journal version: *IEEE Transactions on Information Theory*, 50:7, pp. 1536-1539, 2004.
- “Fine-Grained Layered Multicast,” with J. Byers and M. Luby. In *Proceedings of IEEE INFOCOM 2001*, pp. 1143-1151, 2001. Journal version: “Fine-Grained Layered Multicast with STAIR,” with J. Byers, Gu-In Kwon, and M. Luby. To appear in *IEEE/ACM Transactions on Networking*.
- “Using Multiple Hash Functions to Improve IP Lookups,” with A. Broder. In *Proceedings of IEEE INFOCOM 2001*, pp. 1454-1463, 2001.
- “Improved Results for Route Planning in Stochastic Transportation Networks,” with J. Boyan. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 895-902, 2001.

“Estimating Resemblance of MIDI Documents,” with S. Owen. In *Proceedings of the 3rd Workshop on Algorithm Engineering and Experiments*, available as *Lecture Notes in Computer Science 2153*, pp. 78-90, 2001.

“Linear Waste of Best Fit Bin Packing on Skewed Distributions,” with Claire Kenyon. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pp. 582-589, 2000. Journal version: invited to special issue of *Random Structures and Algorithms*, 20(3), pp. 441-464, 2002.

“FLID/DL: Congestion Control for Layered Multicast,” with J. Byers, M. Frumin, G. Horn, M. Luby, A. Roetter, and W. Shaver. In *Proceedings of the 2nd International Workshop on Networked Group Communication (NGC 2000)*, pp. 71-81, 2000. Journal version: with J. Byers, G. Horn, M. Luby, and W. Shaver. *IEEE Journal on Selected Areas in Communications*, 20:8, pp. 1558-1570, October 2002.

“On Near-Uniform URL Sampling,” with M. Henzinger, A. Heydon, and M. Najork. In *Proceedings of the Ninth International World Wide Web Conference*, pp. 295-308, 2000.

“An Extension of Path Coupling and Its Application to the Glauber Dynamics for Path Coupling,” with M. Dyer, L. Goldberg, C. Greenhill, and M. Jerrum. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 616-623, 2000. Journal version: *SIAM Journal on Computing*, 30(6), pp. 1962-1975, 2001.

“Improved Classification via Connectivity Information,” with A. Broder and R. Krauthgamer. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 576-585, 2000.

“The Asymptotics of Selecting the Shortest of Two, Improved,” with B. Vöcking. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, pp. 326-327, 1999. Extended version appears as a chapter in *Analytic Methods in Applied Probability: In Memory of Fridrikh Karpelevich*, American Mathematical Society, pp. 165-176, 2002.

“Completeness and Robustness Properties of Min-Wise Independent Permutations,” with A. Broder. In *Proceedings of Random '99*, available as *Lecture Notes in Computer Science 1671*, pp. 1-10, 1999. Journal version: *Random Structures and Algorithms*, 18(1), pp. 18-30, January 2001.

“Measuring Index Quality Using Random Walks on the Web,” with M. Henzinger, A. Heydon, and M. Najork. In *Proceedings of the Eighth International World Wide Web Conference*, pp. 213-225, Elsevier Science, 1999.

“Analysis of Timing Based Mutual Exclusion with Random Times,” with E. Gafni. In *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing*, pp. 13-21, 1999. Journal version: *SIAM Journal on Computing*, 31:3, pp. 816-837, 2001.

“Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads,” with J. Byers and M. Luby. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99)*, pp. 275-284, 1999.

“Unscrambling Address Lines (Abstract),” with A. Broder and L. Moll. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 870-871, 1999.

“A Derandomization Using Min-Wise Independent Permutations,” with A. Broder and M. Charikar. In *Proceedings of Random '98*, available as *Lecture Notes in Computer Science 1518*, pp. 15-24, October 1998. Journal version: *Journal of Discrete Algorithms*, 1:1, pp. 11-20, 2003.

“On Balls and Bins with Deletions,” with R. Cole, A. Frieze, B. M. Maggs, A. W. Richa, R. Sitaraman, and E. Upfal. In *Proceedings of Random '98*, available as *Lecture Notes in Computer Science 1518*, pp. 145-158, October 1998.

“Studying Balanced Allocations with Differential Equations.” *Combinatorics, Probability, and Computing*, vol. 8, pp. 473-482, 1999.

“Delayed Information and Action in On-Line Algorithms,” with S. Albers and M. Charikar. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS-98)*, pp. 71-80, November 1998. Journal version: *Information and Computation*, 170, pp. 135-152, 2001.

“A Digital Fountain Approach to the Reliable Distribution of Bulk Data,” with John Byers, Michael Luby, and Ashu Rege. In *Proceedings of ACM SIGCOMM '98*, available as *Computer Communications Review*, vol. 28:4, pp. 56-67, 1998. Journal version: *IEEE Journal on Selected Areas in Communication*, 20:8, pp. 1528-1540, October 2002.

“Analyses of Load Stealing Models Using Differential Equations.” In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 212-221, 1998. Journal version: *Theory of Computing Systems*, 34:1, pp. 77-98, January 2001.

“Load Adaptive Routing on Butterfly Networks,” with R. Cole, B. Maggs, F. Meyer auf der Heide, A. Richa, K. Schröder, R. Sitaraman, and B. Vöcking. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pp. 378-388, ACM Press, May 23-26, 1998.

“Min-Wise Independent Permutations,” with Andrei Broder, Moses Charikar, and Alan Frieze. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pp. 327-336, ACM Press, May 23-26, 1998. Journal version: *Journal of Computer and System Sciences*, 60(3), pp. 630-659, 2000.

“Analysis of Low Density Codes and Improved Designs Using Irregular Graphs,” with Michael Luby, Amin Shokrollahi, and Dan Spielman. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pp. 249-258, ACM Press, May 23-26, 1998.

“Improved Low Density Parity Check Codes Using Irregular Graphs and Belief Propagation,” with Michael Luby, Amin Shokrollahi, and Dan Spielman. In *Proceedings of the 1998 International Symposium on Information Theory*. Extended version available as Compaq Systems Research Center Technical Note 1998-009. Journal version: “Improved Low Density Parity Check Codes Using Irregular Graphs,” *IEEE Transactions on Information Theory*, 47(2), pp. 585-598, 2001.

“Analysis of Random Processes via And-Or Tree Evaluation,” with Michael Luby and Amin Shokrollahi. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 364-373, 1998.

“Average Case Analyses of First Fit and Random Fit Bin Packing,” with Susanne Albers. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp.

290-299, 1998. Journal version: *Random Structures and Algorithms*, 16(3), pp. 240-259, 2000.

“How Useful is Old Information?” In *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing*, pp. 83-91, 1997. Journal version: *IEEE Transactions on Parallel and Distributed Systems*, 11:1, pp. 6-20, January 2000.

“Revisiting the COUNTER Algorithms for List Update,” with Susanne Albers. *Inf. Proc. Letters*, 64, pp. 155-160, 1997.

“Practical Loss-Resilient Codes,” with Michael Luby, Amin Shokrollahi, Dan Spielman, and Volker Stemann. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pp. 150-159, 1997. Journal version: “Efficient Erasure Correcting Codes,” with Michael Luby, Amin Shokrollahi, and Dan Spielman. *IEEE Transactions on Information Theory*, 47(2), pp 569-584, 2001.

“On the Analysis of Randomized Load Balancing Schemes.” In *Proceedings of the 9th ACM Symposium on Parallel Algorithms and Architectures*, pp. 292-301, 1997. Journal version: special conference issue of *Theory of Computing Systems*, vol. 32, pp. 361-386, 1999.

“Load Balancing and Density Dependent Jump Markov Processes.” In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pp. 213-222, 1996. Journal version: “The Power of Two Choices in Randomized Load Balancing.” *IEEE Transactions on Parallel and Distributed Computing*, 12(10), pp. 1094-1104, 2001.

“Designing Stimulating Programming Assignments for an Algorithms Course: A Collection of Problems Based on Random Graphs.” In *SIGCSE Bulletin*, 28:3, pp. 29-36, September 1996.

“Constant Time per Edge is Optimal in Rooted Tree Networks.” In *Proceedings of the 8th ACM Symposium on Parallel Algorithms and Architectures*, pp. 162-169, 1996. Journal version: *Distributed Computing* vol. 10:4, 1997, pp. 189-197.

“Average Case Analyses of List Update Algorithms, with Applications to Data Compression,” with Susanne Albers. In *Proceedings of the 1996 International Colloquium on Automata, Languages, and Programming*, pp. 514-525. Journal version: *Algorithmica*, vol. 21:3, 1998, pp. 312-329.

“Pattern-based Compression of Text Images,” with Andrei Broder. In *1996 Data Compression Conference*, pp. 300-309.

“Parallel Randomized Load Balancing,” with Micah Adler, Soumen Chakrabarti, and Lars Rasmussen. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pp. 238-247, 1995. Journal version: *Random Structures and Algorithms*, vol. 13:2, September 1998, pp. 159-188.

“Bounds on the Greedy Routing Algorithm in Array Networks.” In *Proceedings of the 6th ACM Symposium on Parallel Algorithms and Architectures*, pp. 346-353, 1994. Journal version: special conference issue of the *Journal of Computer and System Sciences* vol. 53:3, December 1996, pp. 317-327.

“Computational Complexity of Loss Networks,” with Graham Louth and Frank Kelly. *Theoretical Computer Science*, vol. 125, May 1994, pp. 45-59.

Other Publications

“Lyra: An Efficient and Expressive Subquadratic Architecture for Modeling Biological Sequences,” with Krithik Ramesh, Sameed Siddiqui, Albert Gu, and Pardis Sabeti. Available at <https://arxiv.org/abs/2503.16351>.

“Juniper: Reconstructing Transmission Events from Next-Generation Sequencing Data at Scale,” with Ivan Specht, Gage Moreno, Taylor Brock-Fisher, Lydia Kraslinkova, Brittany Petros, Jonathan Pekar, Mark Schifferli, Ben Fry, Catherine Brown, Lawrence Madoff, Megan Burns, Stephen Schaffner, Daniel Park, Bronwn MacInnis, Al Ozonoff, Patrick Varilly, and Pardis Sabeti. Available at <https://www.medrxiv.org/content/10.1101/2025.03.02.25323192v1>.

“Delphy: scalable, near-real-time Bayesian phylogenetics for outbreaks,” with Patrik Varilly, Mark Schifferli, Katherine Yang, Tim Burcham, Paul Cronan, Olivia Glennon, Olivia Jacks, Ellory Laning, Libby Marrs, Kyle Oba, Shannon Yeung, Edyth Parker, Ifeanyi Omah, Jonathan E. Pekar, Laura Luebbert, Kristian G. Andersen, Daniel J. Park, Stephen F. Schaffner, Bronwyn L. MacInnis, Christian Happi, Jacob E. Lemieux, Al Ozonoff, Michael D. Mitzenmacher, Ben Fry, and Pardis C. Sabeti. Available at <https://www.biorxiv.org/content/10.1101/2025.03.25.645253v1>.

“Use of popularity information to reduce risk posed by guessing attacks,” with Stuart Schechter and Cormac Herley. US Patent 08959644. Issued 2/17/2015.

“Method and apparatus providing network coding based flow control,” with Jay Kumar Sundararajan, Devavrat Shah, Muriel Medard, Joao Barros, and Szymon Jakubczak. US Patent 08526451. Issued 9/3/2013.

“Method and apparatus providing network coding based flow control,” with Jay Kumar Sundararajan, Devavrat Shah, Muriel Medard, Joao Barros, and Szymon Jakubczak. US Patent 08130776. Issued 3/6/2012.

“Method for finding optimal paths using a stochastic network model,” with Evdokia Nikolova and Matthew Brand. US Patent 07573866. Issued 8/11/2009.

“Ranking search engine results,” with Monika Henzinger. US Patent 07541388. Issued 11/11/2008.

“Distributed, compressed Bloom filter Web cache server.” US Patent 06920477. Issued 7/19/2005.

“Method for packing rectangular strips,” with Neal Lesh and Joe Marks. US Patent 06832129. Issued 12/14/2004.

“System and method for near-uniform sampling of web page addresses,” with Monika Henzinger, Allan Heydon, and Marc Najork. US Patent 06594694. Issued 7/15/2003.

“On demand encoding with a window,” with Michael Luby, Gavin Horn, Jeffrey Persch, John Byers, and Armin Haken. US Patent 06486803. Issued 11/26/2002.

“Generating high weight encoding symbols using a basis,” with Armin Haken, Michael Luby, Gavin Horn, Diane Hernek, and John Byers. US Patent 06411223. Issued 6/25/2002.

“Space-efficient multi-cycle barrel shifter circuit,” with Laurent Moll. US Patent 06314156. Issued 11/6/2001.

“Method for determining a random permutation of variables by applying a test function,” with Andrei Broder, Laurent Moll, and Mark Shand. US Patent 06292762. Issued 9/18/2001.

“Loss resilient code with double heavy tailed series of redundant layers,” with Michael Luby. US Patent 06195777. Issued 2/27/2001.

“Message encoding with irregular graphing,” with Michael Luby, Amin Shokrollahi, Dan Spielman, and Volker Stemann. US Patent 06163870. Issued 12/19/2000.

“Compression of grey scale images of text,” with Andrei Broder. US Patent 06088309. Issued 7/11/2000.

“Irregularly graphed encoding technique,” with Michael Luby, Amin Shokrollahi, Dan Spielman, and Volker Stemann. US Patent 06081909. Issued 6/27/2000.

“Loss resilient decoding technique,” with Michael Luby, Amin Shokrollahi, Dan Spielman, and Volker Stemann. US Patent 06073250. Issued 6/6/2000.

“Task processing optimization in a multiprocessor system,” with Andrei Broder. US Patent 05991808. Issued 11/23/1999.

“Compression protocol with multiple preset dictionaries,” with Andrei Broder and Jeff Mogul. US Patent 05953503. Issued 9/14/1999.

“When Simple Hash Functions Suffice,” with K. Chung and S. Vadhan. Appears as a chapter in *Beyond Worst Case Analysis*, edited by Tim Roughgarden, Cambridge University Press.

“Algorithms with Predictions,” with S. Vassilvitskii. Appears as a chapter in *Beyond Worst Case Analysis*, edited by Tim Roughgarden, Cambridge University Press.

“Some Practical Randomized Algorithms and Data Structures,” with E. Upfal. Appears as a chapter in *Computing Handbook, Third Edition: Computer Science and Software Engineering*, edited by T. Gonzalez, J. Diaz-Herrera, and A. Tucker. Chapman and Hall, 2014.

“Building an Efficient Hash Table on the GPU,” with D. Alcantara, V. Volkov, S. Sengupta, J. Owens, and N. Amenta. Appears as a chapter in *GPU Computing Gems: Jade Edition*, edited by Wen-mei W. Hwu. Morgan Kaufmann, 2011.

“Codes – Protecting Data Against Loss and Errors.” Appears as a chapter in *Algorithms Unplugged*, edited by B. Vöcking, M. Dietzfelbinger, R. Reischuk, and H. Vollmer. Springer, 2010.

“Hash-Based Techniques for High-Speed Packet Processing”, with A. Kirsch and G. Varghese. Appears as a chapter in *Algorithms for Next Generation Networks*, edited by G. Cormode and M. Thottan. Springer, 2010.

“Bloom Filters.” *Encyclopedia of Database Systems*, 2009.

“Unbiasing Random Bits.” *Dr. Dobbs Journal*, No. 335, pp. 101-104, April 2002.

“Good Hash Tables & Multiple Hash Functions.” *Dr. Dobbs Journal*, No. 336, May 2002.

“Challenging Students with Creative Assignments.” *SIGACT News*, 32(1):70-73, 2001.

“An Experimental Assignment on Random Processes.” *SIGACT News*, 32(1):74-78, 2001.

“The Power of Two Random Choices: A Survey of Techniques and Results,” with A. Richa and R. Sitaraman. Appears as a chapter in *Handbook of Randomized Computing: Volume 1*, edited by P. Pardalos, S. Rajasekaran, and J. Rolim, pp. 255-312.

“Tornado Codes for Forward Error Correction,” with Michael Luby, in *Digital Forefront*.

“Bounds and Improvements for BiBa Signature Schemes,” with A. Perrig. Harvard Computer Science Group Technical Report TR-02-02.

“A Note on Low Density Parity Check Codes for Erasures and Errors.” Digital SRC Technical Note 1998-017.

“Tight Thresholds for the Pure Literal Rule.” Digital SRC Technical Note 1997-011.

“The Power of Two Choices in Randomized Load Balancing.” Ph.D. thesis, U.C. Berkeley, August 1996.

“*Calculus: A Solution Manual*,” with others, Harcourt Brace Jovanovich, 1993.

“*Elliptic Curves in Computer Science: Primality Testing, Factoring, and Cryptography*,” senior honors thesis, Harvard University, 1991. Published as a Hoopes Prize winner.

**Conference
Presentations**

“Queueing, Predictions, and LLMs: Challenges and Open Problems.” Tutorial, SIGMETRICS. June 2025.

“SkipPredict: When to Invest in Predictions for Scheduling. Presented at the TTIC Workshop on Learning-Augmented Algorithms, August 2024.

“Algorithms with Predictions.” Invited talk presented at the INFORMS 2023 Annual Meeting, October 2023.

“Algorithms with Predictions.” Presented at the 2nd Vienna Workshop on Computational Optimization. March 2023.

“Algorithms with Predictions.” Presented at the Sublinear Algorithms Workshop, FODSI/MIT. August 2022.

“Algorithms with Predictions: State of the Area.” Presented at the Algorithms with Predictions Workshop at the 54th Annual ACM Symposium on Theory of Computing. June 2022. Also panel lead.

“Efficient Randomized Network Telemetry Algorithms.” Presented at the Data plane algorithms in programmable networks tutorial at ACM SIGMETRICS 2022.

“Scheduling with Predictions and the Price of Misprediction.” Presented at the 11th Annual Innovations in Theoretical Computer Science Conference. January 2020.

“Learning-Augmented Algorithms: How ML Can Lead to Provably Better Algorithms.” ALGO Keynote Talk, September 2019.

“Robust Set Reconciliation via Locality Sensitive Hashing.” Presented at the 38th Symposium on Principles of Database Systems, July 2019.

“Arithmetic Progression Hypergraphs: Examining the Second Moment Method.” Presented at the 2019 ANALCO Workshop, January 2019.

“A Model for Learned Bloom Filters and Optimizing by Sandwiching.” Presented at the NeurIPS 2018 Poster Session.

“Adaptive Cuckoo Filters.” Presented at the 2018 ALENEX Conference, January 2018.

“Analyzing Distributed Join-Idle-Queue: A Fluid Limit Approach.” Presented at the 54th Annual Allerton Conference on Communication, Control, and Computing, October 2016.

“Voronoi Choice Games.” Presented at the 2016 International Colloquium on Automata, Languages, and Programming, July 2016.

“A New Approach to Analyzing Robin Hood Hashing.” Presented at the 2016 ANALCO Workshop, January 2016.

“More Analysis of Double Hashing for Balanced Allocations.” Presented at the 2016 ANALCO Workshop, January 2016.

“Data exploration: equitability and the maximal information coefficient.” Presented at the Workshop for Data-driven Algorithmics, September 2015.

“Multi-Party Set Reconciliation Using Characteristic Polynomials.” Presented at the 2014 Allerton Conference, October 2014.

“Parallel Peeling Algorithms.” Presented at the 2014 Symposium on Parallelism in Algorithms and Architectures, June 2014.

“Balanced Allocations and Double Hashing.” Presented at the 2014 Symposium on Parallelism in Algorithms and Architectures, June 2014.

“Peeling Arguments and Double Hashing.” Presented at the 2012 Allerton Conference, October 2012.

“The Complexity of Object Reconciliation and Open Problems Related to Set Difference and Coding.” Presented at the 2012 Allerton conference, October 2012.

“The Ecology of Communication, Control, and Computing Research and Education (Panel).” Presented at the 2012 Allerton conference, October 2012.

“Peeling Arguments: Invertible Bloom Lookup Tables and Biff Codes.” Presented at the 2012 Workshop on Algorithms for Modern Massive Data Sets (MMDS), July 2012.

“Invertible Bloom Lookup Tables.” Presented at the 2011 Allerton conference, September 2011.

“Brief Announcement: Large-Scale Multimaps.” Presented at the 2011 Symposium on Parallel Algorithms and Architectures, June 2011.

“On the Zero-Error Capacity Threshold for Deletion Channels.” Presented at the 2011 ITA Workshop, February 2011.

“Some Recent Uses of Hashing for Network Applications.” Presented at the 2010 Workshop on Research and Analysis of Tail Phenomena, August 2010.

“Some Recent Uses of Hashing for Network Applications.” Presented at the 2010 Workshop on Random Graphs and Algorithms, June 2010.

“Tight Asymptotic Bounds for the Deletion Channel with Small Deletion Probabilities.” Presented at the 2010 International Symposium on Information Theory, June 2010.

“Some Open Questions on Cuckoo Hashing.” Presented at the 2009 European Symposium on Algorithms, August 2009.

“On the Performance of Multiple Choice Hash Tables with Moves on Deletions and Inserts.” Presented at the 2008 Allerton conference, September 2008.

“Designing Floating Codes for Expected Performance.” Presented at the 2008 Allerton conference, September 2008.

“A Survey of Results for Deletion Channels and Related Synchronization Channels.” Presented at the 11th Scandinavian Workshop on Algorithm Theory, July 2008.

“Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream.” Presented at the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, January 2008.

“Using a Queue to De-amortize Cuckoo Hashing in Hardware.” Presented at the 2007 Allerton conference, September 2007.

“Capacity Upper Bounds for Deletion Channels.” Presented at the 2007 Int’l Symposium on Information Theory (ISIT), June 2007.

“Codes for Deletion and Insertion Channels with Segmented Errors.” Presented at the 2007 Int’l Symposium on Information Theory (ISIT), June 2007.

“Digital Fountains: Applications and Related Issues.” Presented at the INFORMS Annual Meeting, November 2006.

“New Directions for Power Law Research.” Presented at the Radcliffe Exploratory Seminar on Dynamic Networks, October 2006.

“Bloom Filters via d -left Hashing and Dynamic Bit Reassignment.” Presented at the 2006 Allerton conference, September 2006.

“Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines.” Presented at the 2006 SIGCOMM conference, September 2006.

“An Improved Construction for Counting Bloom Filters.” Presented at the European Symposium on Algorithms, September 2006.

“Building a Better Bloom Filter.” Presented at the European Symposium on Algorithms, September 2006.

“On the Theory and Practice of Data Recovery with Multiple Versions.” Presented at the 2006 International Symposium on Information Theory, July 2006.

“New Results for an Old Data Structure: Bloom Filters and Beyond.” Presented at the 2006 Stochastic Networks conference, June 2006.

“New Results and Open Problems for Deletion Channels.” Presented at ANALCO 2006, January 2006 (plenary talk).

“Digital Fountains and their Application to Informed Content Delivery over Adaptive Overlay Networks.” Presented at 19th International Symposium on Distributed Computing, September 2005 (plenary talk).

“Simple Summaries for Multiple Choice Hashing.” Presented at the 2005 Allerton Conference.

“New Directions in Power Law Research.” MSRI Workshop on Models of Real-World Random Networks, April 2005.

“Multi-dimensional Balanced Allocations.” Presented at the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, January 2005.

“Digital Fountains: A Survey and Look Forward.” Presented at the 2004 Information Theory Workshop. October 2004.

“Improved Lower Bounds on the Capacity of I.I.D. Deletion Channels.” Presented at the 2004 Allerton Conference.

“Geometric Generalizations of the Power of Two Choices.” Presented at the 16th ACM Symposium on Parallel Algorithms and Architectures, June 2004.

“New Models and Methods for File Size Distributions.” Presented at the 40th Annual Allerton Conference on Communication, Control, and Computing, October 2003.

“Verification Codes for Deletion Channels.” Presented at the International Symposium on Information Theory, July 2003.

“New Exhaustive, Heuristic, and Interactive Algorithms for 2-Dimensional Bin-Packing.” Presented at the 2nd Annual Workshop on Random Graphs and Algorithms, June 2003.

“Network Applications of Bloom Filters: A Survey.” Presented at the 40th Annual Allerton Conference on Communication, Control, and Computing, October 2002.

“Verification Codes.” Presented at the 40th Annual Allerton Conference on Communication, Control, and Computing, October 2002.

“New Directions in Balls and Bins Problems.” Invited talk for the 6th International Workshop on Randomization and Approximation Methods in Computer Science. September 2002.

“Optimal Plans for Aggregation.” Presented at the 21st Annual ACM Symposium on Principles of Distributed Computing, July 2002.

“How to Get Across Town by Bus.” Presented at the INFORMS Annual Meeting, November 2001.

“A Brief History of Lognormal and Power Law Distributions.” Presented at the 39th Annual Allerton Conference on Communication, Control, and Computing, October 2001.

“Compressed Bloom Filters.” Presented at the 20th Annual ACM Symposium on Principles of Distributed Computing, August 2001.

“How Useful Is Old Information?” Presented at the INFORMS Applied Probability Conference, July 2001.

“Using Multiple Hash Functions to Improve IP Lookups.” Presented at the INFOCOM, April 2001.

“On the Hardness of Finding Multiple Preset Dictionaries.” Presented at the 2001 Data Compression Conference, March 2001.

“Improved Results for Route Planning in Stochastic Transportation Networks.” Presented at the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, January 2001.

“Estimating Resemblance of MIDI Documents.” Presented at the 3rd Workshop on Algorithm Engineering and Experiments, January 2001.

“Linear Waste of Best Fit Bin Packing on Skewed Distributions.” Presented at the 41st Annual IEEE Symposium on Foundations of Computer Science, November 2000.

“Improved Classification via Connectivity Information.” Presented at the 11th Annual ACM Symposium on Discrete Algorithms, January 2000.

“The Asymptotics of Selecting the Shortest of Two, Improved.” Presented at the 37th Annual Allerton Conference on Communication, Control, and Computing, August 1999.

“Analysis of Timing Based Mutual Exclusion with Random Times.” Presented at the 18th Annual ACM Symposium on Principles of Distributed Computing, May 1999.

“Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads.” Presented at INFOCOM ’99, March 1999.

“Delayed Information and Action in On-Line Algorithms.” Presented at the 39th Annual IEEE Symposium on Foundations of Computer Science, November 1998.

“A Derandomization Using Min-Wise Independent Permutations.” Presented at Random ’98, October 1998.

“On Balls and Bins with Deletions.” Presented at Random ’98, October 1998.

“The Power of Two Choices and Other Examples of Using Differential Equations to Analyze Algorithms.” Invited presentation at the Warwick Randomized Algorithms and Stochastic Simulation Tutorial and Workshop. July 1988.

“Analysis of Low Density Codes and Improved Designs Using Irregular Graphs.” Presented at the 30th Annual ACM Symposium on Theory of Computing, May 1998.

“Average Case Analyses of First Fit and Random Fit Bin Packing.” Presented at the 9th Annual ACM Symposium on Discrete Algorithms, January 1998.

“How Useful Is Old Information?” Presented at the 16th Annual ACM Symposium on Principle of Distributed Computing, August 1997.

“On the Analysis of Randomized Load Balancing Schemes.” Presented at the 9th ACM Symposium on Parallel Algorithms and Architectures, Newport, Rhode Island, June 1997.

“Load Balancing and Density Dependent Jump Markov Processes.” Presented at the 37th Annual Symposium on Foundations of Computer Science, Burlington, Vermont, October 1996.

“Constant Time per Edge is Optimal in Rooted Tree Networks.” Presented at the 8th ACM Symposium on Parallel Algorithms and Architectures, Padua, Italy, June 1996.

“GRID: Pattern-based Compression of Text Images.” Presented at the 1996 Data Compression Conference, Snowbird, Utah, April 1996.

“Parallel Randomized Load Balancing.” Presented at the 27th ACM Symposium on the Theory of Computing, Las Vegas, Nevada, May 1995. Extended version presented at the Bay Area Theory Seminar, July 1995.

“Bounds on the Greedy Routing Algorithm on Array Networks.” Presented at the 6th ACM Symposium on Parallel Algorithms and Architectures, Cape May, New Jersey, June 1994.

Other Presentations

“Algorithms with Predictions in Scheduling.” Johns Hopkins University. 2025.

“Algorithms with Predictions.” Georgia Tech. 2023.

“Modern Algorithms Lecture Series.” Lipari Summer School. 2022.

“Algorithms with Predictions.” Carnegie Mellon, UC Santa Barbara, Technical University of Munich. 2022.

“Locality Sensitive Hashing.” Broad Institute (Models, Inference, and Algorithms).

“Algorithms with Predictions.” Open Data Science Conference East 2020, Open Data Science Conference Europe 2020, Foundations of Data Science Institute Kickoff Workshop 2020.

“Learning-Augmented Algorithms: How ML Can Lead to Provably Better Algorithms.” Rice University. 2019.

“Metric Sublinear Algorithms via Linear Sampling.” Sublinear Algorithms Workshop (MIT), Baidu Research. 2018.

“Bloom Filters, Cuckoo Hashing, Cuckoo Filters, and Adaptive Cuckoo Filters.” Northeastern University, Univ. of Illinois at Urbana-Champaign, Georgetown University, University of California Irvine, Baidu Research, Georgia Tech. 2018.

“Hashing Lecture Series.” Swedish Summer School in Computer Science. 2016.

“More Analysis of Double Hashing for Balanced Allocations.” U.C. Irvine. 2016.

“Algorithm Engineering with Hashing,” U.C. Irvine, AMD. 2016.

“Bloom Filters and Such,” EADS Summer School on Hashing: Theory and Applications, July 2014.

“Cuckoo Hashing, Balanced Allocations, and Such,” EADS Summer School on Hashing: Theory and Applications, July 2014.

“Peeling Arguments: Invertible Bloom Lookup Tables and Biff Codes,” ICERM Workshop on Stochastic Graph Models, March 2014.

“Peeling Arguments: Invertible Bloom Lookup Tables and Biff Codes,” University of Copenhagen. 2013.

“Some Hash-Based Data Structures and. Algorithms Everyone Should Know.” UC Davis, 2013.

“Peeling Algorithms,” Simons Institute. 2013.

“Verifying Computations in the Cloud (and Elsewhere),” TTI/Vanguard : Ginormous Systems. Microsoft Faculty Summit. 2013.

“The Groupon Effect on Yelp Ratings: a Root Cause Analysis,” Cornell, 2012.

“Peeling Arguments: Invertible Bloom Lookup Tables and Biff Codes,” Google. 2012.

“Detecting Novel Associations in Large Data Sets.” eHarmony (LA Machine Learning). 2012.

“Cuckoo Hashing: New Results and Open Questions.” University of Madison-Wisconsin. 2011.

“Practical Verified Computation with Streaming Interactive Proofs.” University of Madison-Wisconsin. 2011.

“The Power of Choice: Simple Hash-Based Data Structures Everyone Should Know.” Santa Fe Institute. 2011.

“Information Asymmetries in Pay-Per-Bid Auctions: How Swoopo Makes Bank.” Liverpool University, Microsoft UK. 2010.

“Some Open Questions on Cuckoo Hashing.” Stanford, Microsoft Research, Yale, Dartmouth, Princeton, Cambridge University, University College London . 2009-2010.

“An Efficient, Rigorous Approach for Identifying Statistically Significant Frequent Itemsets.” Google. 2009.

“Some Recent Results on Cuckoo Hashing.” AT&T. 2009.

“Some Results on Coding for Flash Memory.” University College London. 2009.

“Network Coding Meets TCP.” University College London. 2009.

“A Survey of Results for Deletion Channels and Related Synchronization Channels.” Cambridge, Georgia Tech, Microsoft Research, UCLA, 2009-2010.

“Bloom Filters, Related Data Structures, and their Applications.” Cambridge (Kuwait Lecture). 2009.

“A History of and New Directions for Power Law Research.” SUNY Buffalo. 2009.

“Why Simple Hash Functions Work.” Cisco. 2008.

“Cuckoo Hashing and CAMs.” Cisco, Google. 2008.

“New Results in Hash-Based Data Structures.” Cisco. 2007.

“The Hiring Problem and Lake Wobegon Strategies.” Microsoft, U.C. Berkeley, Princeton, Google, Yahoo. 2007-2008.

“A Brief History of Lognormal and Power Law Distributions.” Harvard Statistics. 2006.

“New Results and Open Problems for Deletion Channels and Related Synchronization Problems.” CMU, Brown, Harvard. 2006.

“Some New Results on an Old Data Structure, Bloom Filters.” Stanford, Cisco, Microsoft. 2005.

“Improved Lower Bounds for Deletion Channels.” MIT, EPFL (Lausanne). 2005.

“Building a Better Bloom Filter.” Google, Tufts, EPFL (Lausanne), MIT. 2004-05.

“Geometric Generalizations of the Power of Two Choices.” Columbia, Brown, MIT, Georgia Tech. 2004-05.

“New Lower Bounds on the Capacity of the Binary Deletion Channel.” MIT, 2004.

“Digital Fountains, and their Use for Informed Content Delivery across Overlay Networks.” Cornell, 2003.

“Verification Codes for Error and Deletion Channels.” U.C. Berkeley, 2003.

“New Exhaustive, Heuristic, and Interactive Algorithms for 2D Strip Packing.” U.C. Berkeley, 2003.

“Dynamic Models for File Sizes and Double Pareto Distributions.” IBM Research, Microsoft Silicon Valley, Stanford, Tufts, Boston University, Random Graphs and Algorithms Workshop. 2002.

“Verification Codes.” DIMACS workshop. 2001.

“Getting Across Town by Bus, Aggregation, and Reliability.” MIT (LIDS), Mitsubishi Electronic Research Labs. 2001.

“New Ideas for Layered Multicast Congestion Control.” International Computer Science Institute. 2001.

“Compressed Bloom Filters, and Towards Compressing Web Graphs.” U. of Washington, AT&T Labs, U. C. Berkeley, MIT, Brandeis, Carnegie Mellon University, Stanford, DIMACS workshop on compression, U.C. San Diego. 2001.

“Toward Compressing Web Graphs.” Compaq Systems Research Center, Altavista Corporation. 2000.

“Improved Results for Route Planning in Stochastic Transportation Networks.” Compaq Systems Research Center. 2000.

“Choosing the Shortest of Two, Improved, with Applications to IP Routing.” U. Washington, Princeton, Yale, U.C. Berkeley, Boston University, Stanford, M.I.T., Northeastern, Compaq Systems Research Center. 2000.

“Tornado Codes, with Applications to Reliable Multicast.” Harvard, CMU, Duke, AT&T Labs, Univ. of Illinois at Urbana-Champaign, University of Rhode Island, Boston University. 1998-2000.

“How Useful Is Old Information?” MIT, U.C. Berkeley, Stanford, Microsoft. 1997.

“Practical Loss-Resilient Codes.” U.C. Berkeley, Harvard. 1997.

“An Average Case Analysis for First-Fit Bin Packing.” Stanford, DIMACS workshop. 1997.

“The Power of Two Choices.” U.C. Berkeley, Brown, Harvard, Boston University, IBM Almaden, Digital Systems Research Center. 1997.

“The Power of Two Choices and Other Examples of Using Differential Equations to Analyze Algorithms.” U. Mass.