

Paper No. ____

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TESSELL, INC.,
Petitioner,

v.

NUTANIX, INC.,
Patent Owner.

Case No. IPR2025-00322
Patent No. 11,010,336

**PETITION FOR INTER PARTES REVIEW
UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.1 et seq**

TABLE OF CONTENTS

MANDATORY NOTICES.....	xii
A. Real Party-In-Interest	xii
B. Related Matters.....	xii
1. United States Patent & Trademark Office	xii
2. U.S. District Court for the Northern District of California.....	xii
C. Counsel and Service Information - § 42.8(b)(3) and (4).....	xii
I. INTRODUCTION	1
II. STANDING CERTIFICATION.....	6
III. GROUNDS FOR UNPATENTABILITY	6
IV. THE '336 PATENT.....	7
A. Described Embodiments.....	7
B. Prosecution History	11
C. Person of Ordinary Skill in the Art (“POSA”).....	11
V. CLAIM INTERPRETATION	12
VI. <u>GROUND 1</u> : Claims 1, 5-8, 10-15, 19-21 Are Unpatentable Over Sivasubramanian+Zha	12
A. Sivasubramanian (EX1005).....	12
1. Overview	12
a. Database-Provisioning UIs	16
i. GUI	17
ii. Command-Line	21
b. Backup “Workflow”	23
2. Motivation to Customize Sivasubramanian’s UI	24
B. Zha (EX1006).....	25
C. Sivasubramanian+Zha Combination	27
1. UI for retention period.....	32
2. UI for protection schedule.....	35
D. Claim-by-Claim Analysis.....	36

1. Claim 1	36
a. [1PRE]: “A method comprising”	36
b. [1A1]: “receiving, by a processor executing computer-readable instructions stored on a memory...”	36
c. [1A2]: “a user request for provisioning a source database...”	38
i. GUI	38
ii. Command Line	40
d. [1A3]: “in a hyperconverged infrastructure system”	41
e. [1B]: “receiving, by the processor, selection of a database engine type”	43
i. GUI	43
ii. Command Line	45
f. [1C]: “receiving, by the processor, selection of a Service Level Agreement (“SLA”) and a protection schedule”	45
i. GUI	45
ii. Command Line	46
g. [1D]: “provisioning, by the processor, the source database based upon the database engine type; and”	46
h. [1E]: “setting up, by the processor, automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database”	47
i. [1F]: “wherein the transactional logs record transactions occurring on the source database since a most recent snapshot”	50
j. [1G]: “and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.”	51
2. Claim 5	51
3. Claim 6	52
4. Claim 7	54

5. Claim 8	54
a. [8PRE]-[8A].....	54
b. [8B]	55
6. Claim 10	55
7. Claim 11	56
8. Claim 12	58
a. [12PRE].....	58
b. Limitations [12A1]-[12G].....	58
9. Claim 13	59
10. Claim 14	59
11. Claim 15	60
12. Claim 19	61
a. [19PRE].....	61
a. Limitations [19A]-[19G].....	62
13. Claim 20	62
14. Claim 21	62
VII. <u>GROUND 2: Claims 1-23 Are Unpatentable Over</u> Sivasubramanian+Shekar+Zha	62
A. Shekar (EX1007).....	62
B. Sivasubramanian+Shekar+Zha Combination.....	64
C. Claim-by-Claim Analysis.....	68
1. Claims 1, 5-8, 10-15, 19-21.....	68
2. Claim 2	68
a. [2PRE]-[2A].....	68
b. [2B]	70
3. Claim 3	72
a. “software profile”.....	72
b. “compute profile”.....	74
c. “network profile”	75

d. “database parameters profile”	76
4. Claim 4	77
5. Claim 9	81
6. Claim 16	82
7. Claim 17	82
8. Claim 18	82
9. Claim 22	83
10. Claim 23	83
VIII.NO BASIS EXISTS FOR DISCRETIONARY DENIAL	83
A. 315(a).....	83
1. Stay Potential.....	83
2. Trial Timing	84
3. Litigation Investment	84
4. Issue overlap.....	84
5. Litigation Defendants.....	85
6. Other Circumstances	85
B. 325(d).....	85
1. Part One	85
2. Part Two	86
IX. CONCLUSION.....	87
X. CLAIM LISTING.....	88

TABLE OF AUTHORITIES

CASES

<i>Advanced Bionics, LLC v. Med-El Elektromedizinische Gerate GMBH</i> , IPR2019-01469, Paper 6 (Feb. 13, 2020).....	86, 88
<i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, Paper 11 (Mar. 20, 2020)	84, 85, 86
<i>Apple v. Seven Networks</i> , IPR2020-00156, Paper 10 (June 15, 2020)	85, 86
<i>Becton, Dickinson & Co. v. B. Braun Melsungen AG</i> , IPR2017-01586, Paper 8 (Dec. 15, 2017)	88
<i>Cellco P'ship v. Huawei Device Co.</i> , IPR2020-01117, Paper 10 (Feb. 3, 2021).....	88
<i>Group III v. Targus</i> , IPR2021-00371, Paper 21 (July 9, 2021).....	89
<i>Sand Revolution II v. Continental Intermodal Group-Trucking</i> , IPR2019-01393, Paper 24 (June 16, 2020)	85, 86

REGULATIONS

37 C.F.R. §42.100(b)	12
37 C.F.R. §42.104(a).....	6

STATUTES

35 U.S.C. §282(b)	12
35 U.S.C. §102(a)	6
35 U.S.C. §103	7
35 U.S.C. §315(a)	84
35 U.S.C. §325(d)	86

OTHER AUTHORITIES

Director’s Interim Procedure for Discretionary Denials (June 21, 2022)	84, 85
PTAB Rules for 325(d) Considerations, 89 Fed. Reg. 28693 (Apr. 19, 2024).....	87, 89

APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 11,010,336
1002	Prosecution History of U.S. Patent No. 11,010,336
1003	Declaration of Dr. Samrat Bhattacharjee (“Bhattacharjee”)
1004	Curriculum Vitae of Dr. Bhattacharjee
1005	U.S. Patent No. 8,713,060 (“Sivasubramanian”)
1006	U.S. Patent No. 8,150,808 (“Zha”)
1007	U.S. Patent App. Pub. No. 2018/0232142 (“Shekar”)
1008	U.S. Patent No. 6,542,893 (“Quernemoen”)
1009	The American Heritage Dictionary of the English Language (5th ed. 2011) (“Heritage”)
1010	U.S. Patent No. 7,873,684 (“Souder”)
1011	U.S. Patent App. Pub. No. 2018/0324279 (“Barton”)
1012	Using Oracle Database Cloud Service – Creating a Database Deployment (2017) (“Oracle”) (Wayback Machine Capture, January 18, 2017), available at https://web.archive.org/web/20170118071307/https://docs.oracle.com/en/cloud/paas/database-dbaascloud/csdbi/create-db-deployment.html
1013	Amazon Relational Database Service User Guide (2016) (“RDS”) (Wayback Machine Capture, April 12, 2016), available at https://web.archive.org/web/20160412105355/http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-ug.pdf
1014	U.S. Patent App. Pub. No. 2009/0319582 (“Simek”)
1015	U.S. Patent No. 7,331,000 (“Ohno”)
1016	U.S. Patent App. Pub. No. 2016/0182328 (“Bhasin”)
1017	U.S. Patent App. Pub. No. 2016/0077925 (“Tekade”)
1018	U.S. Patent App. Pub. No. 2019/0324865 (“Weissman”)
1019	U.S. Patent App. Pub. No. 2016/0335369 (“Picard”)
1020	U.S. Patent No. 11,308,114 (“Moghe”)

Exhibit	Description
1021	U.S. Patent App. Pub. No. 2014/0229698 (“Sivasubramanian-698”)
1022	“Taking a Backup Using Snapshots” (“LVM-Howto”) (Wayback Machine Capture, January 20, 2007), available at https://web.archive.org/web/20070120200004/https://tldp.org/HOWTO/LVM-HOWTO/snapshots_backup.html
1023	U.S. Patent App. Pub. No. 2004/0059878 (“Madany”)
1024	U.S. Patent App. Pub. No. 2006/0161879 (“Lubrecht”)
1025	PCT Pub. No. WO 2010/106578 (“Utsunomiya”)
1026	U.S. Patent No. 10,855,554 (“Freeman”)
1027	U.S. Patent No. 11,182,372 (“Jain”)
1028	U.S. Patent App. Pub. No. 2016/0057077 (“Gomatam”)
1029	U.S. Patent No. 11,036,696 (“Higginson”)
1030	U.S. Patent No. 9,305,068 (“Esposito”)
1031	U.S. Patent No. 7,873,610 (“Poulsen”)
1032	“Creating Instances,” Google Cloud (“Google”) (Wayback Machine Capture, September 18, 2018), available at https://www.archive.org/web/20180918103721/https://cloud.google.com/sql/docs/mysql/create-instance
1033	U.S. Patent App. Pub. No. 2003/0046353 (“Chung”)
1034	U.S. Patent App. Pub. No. 2006/0253472 (“Wasserman”)
1035	U.S. Patent App. Pub. No. 2005/0027661 (“Lober”)
1036	U.S. Patent App. Pub. No. 2018/0173452 (“Hu”)
1037	U.S. Patent No. 10,078,465 (“Wu”)
1038	U.S. Patent No. 9,830,082 (“Srinivasan”)
1039	U.S. Patent App. Pub. No. 2016/0266913 (“Achillopoulos”)
1040	U.S. Patent App. Pub. No. 2017/0262232 (“Zhao”)
1041	U.S. Patent App. Pub. No. 2019/0116237 (“Gibson”)
1042	U.S. Patent No. 10,206,092 (“Mellquist”)
1043	U.S. Patent No. 11,243,703 (“Watkins”)

Exhibit	Description
1044	U.S. Patent App. Pub. No. 2015/0207683 (“Adogla”)
1045	U.S. Patent App. Pub. No. 2014/0018947 (“Ales”)
1046	U.S. Patent App. Pub. No. 2013/0332251 (“Ionnidis”)
1047	U.S. Patent App. Pub. No. 2006/0107006 (“Green”)
1048	U.S. Patent App. Pub. No. 2008/0059894 (“Cisler”)
1049	U.S. Patent App. Pub. No. 2013/0110779 (“Taylor”)
1050	U.S. Patent App. Pub. No. 2016/0306840 (“Deshmukh”)
1051	U.S. Patent App. Pub. No. 2018/0121521 (“Bivol”)
1052	U.S. Patent No. 10,496,302 (“Gaurav”)
1053	U.S. Patent No. 10,268,551 (“Wong”)
1054	U.S. Patent App. Pub. No. 2014/0108339 (“Marsden”)
1055	U.S. Patent No. 8,335,765 (“McAlister”)
1056	Scheduling Order in <i>Nutanix, Inc. v. Tessell, Inc.</i> , Case No. 3:24-cv-01729 (N.D. Cal. Mar. 20, 2024)
1057	Docket Sheet of <i>Nutanix, Inc. v. Tessell, Inc.</i> , Case No. 3:24-cv-01729 (N.D. Cal.)
1058	Affidavit of Nathaniel E. Frank-White
1059	RESERVED
1060	U.S. Patent No. 11,120,011 (“Priebe”)
1061	U.S. Patent No. 7,634,679 (“Quintanillo”)
1062	U.S. Patent App. Pub. No. 2009/0276771 (“Nickolov”)
1063	U.S. Patent App. Pub. No. 2016/0188421 (“Karinta”)
1064	U.S. Patent App. Pub. No. 2017/0243275 (“Goens”)
1065	U.S. Patent App. Pub. No. 2016/0378361 (“Uriel”)
1066	RESERVED
1067	U.S. Patent App. Pub. No. 2017/0228227 (“Winterfeldt”)
1068	U.S. Patent App. Pub. No. 2014/0337844 (“Jin-844”)
1069	U.S. Patent No. 9,582,297 (“Jin-297”)

Exhibit	Description
1070	U.S. Patent No. 10,572,292 (“Bhandari”)
1071	U.S. Patent App. Pub. No. 2015/0188775 (“VanDerWalt”)
1072	U.S. Patent App. Pub. No. 2018/0063088 (“Hardy”)
1073	U.S. Patent No. 4,742,482 (“Inskeep”)
1074	U.S. Patent App. Pub. No. 2015/0207682 (“Nichele”)
1075	U.S. Patent App. Pub. No. 2013/0232497 (“Jalagam”)
1076	U.S. Patent App. Pub. No. 2010/0017801 (“Kundapur”)
1077	U.S. Patent App. Pub. No. 2006/0029096 (“Babbar”)
1078	UK Patent Application No. GB2451943 (“Wren-Hilton”)
1079	U.S. Patent App. Pub. No. 2013/0238331 (“Mikan”)
1080	U.S. Patent App. Pub. No. 2016/0239637 (“Miller”)
1081	U.S. Patent App. Pub. No. 2016/0066201 (“Kerpez”)
1082	U.S. Patent App. Pub. No. 2018/0060119 (“Zamir”)
1083	U.S. Patent App. Pub. No. 2015/0188789 (“Jayaprakash”)
1084	U.S. Patent App. Pub. No. 2015/0207703 (“Gallagher”)
1085	U.S. Patent App. Pub. No. 2006/0161462 (“Sharma”)
1086	RESERVED
1087	U.S. Patent App. Pub. No. 2002/0082858 (“Heddaya”)
1088	<i>Supported Platforms: MySQL Database, available at https://web.archive.org/web/20181106221826/https://www.mysql.com/support/supportedplatforms/database.html (“MySQL”)</i>
1089	U.S. Patent No. 11,061,709 (“Cao”)
1090	U.S. Patent No. 10,705,755 (“Wang”)
1091	U.S. Patent App. Pub. No. 2017/0371547 (“Fruchtman”)
1092	U.S. Patent No. 10,824,522 (“Sadavarte”)
1093	U.S. Patent No. 8,146,080 (“Carter”)
1094	U.S. Patent App. Pub. No. 2009/0276833 (“Paul”)
1095	U.S. Patent App. Pub. No. 2017/0315877 (“Kaplingat”)

Exhibit	Description
1096	U.S. Patent App. Pub. No. 2002/0019920 (“Reuter”)
1097	U.S. Patent App. Pub. No. 2016/0246503 (“Phelan”)
1098	U.S. Patent App. Pub. No. 2017/0068469 (“Shankar”)
1099	U.S. Patent App. Pub. No. 2011/0138383 (“Le”)
1100	U.S. Patent App. Pub. No. 2017/0119753 (“Gong”)
1101	A screenshot of the Expedia webpage displaying the “Preferred class” dropdown menu in its expanded state, Expedia website (Wayback Machine Capture, July 15, 2018), available at https://web.archive.org/web/20180715133951%20https://www.expedia.com/#expand
1102	Excerpts of McGraw-Hil Dictionary of Scientific and Technical Terms (6 th ed. 2003) (“McGraw-Hill”)
1103	U.S. Patent App. Pub. No. 2018/0203771 (“Heidel”)
1104	U.S. Patent App. Pub. No. 2017/0357667 (“Auch”)
1105	Excerpts of New Oxford American Dictionary (1st ed. 2001) (“Oxford”)

MANDATORY NOTICES

A. Real Party-In-Interest

Petitioner is the Real Party-in-Interest.

B. Related Matters

A decision in this proceeding could affect or be affected by the following:

1. United States Patent & Trademark Office

The following U.S. patent applications claim the benefit of priority to U.S. Patent No. 11,010,336:

(i) U.S. Patent Application No. 17/237,599 (U.S. Patent No. 11,604,762), filed April 22, 2021; (ii) U.S. Patent Application No. 18/113,528 (U.S. Patent No. 11,860,818), filed February 23, 2023; (iii) U.S. Patent Application No. 18/489,837 (U.S. Patent No. 12,026,124), filed October 18, 2023; (iv) U.S. Patent Application No. 18/740,432, filed June 11, 2024.

2. U.S. District Court for the Northern District of California

A decision in this proceeding could affect or be affected by the following:

Nutanix, Inc. v. Tessell, Inc., Case No. 3:24-cv-01729 (NDCA).

C. Counsel and Service Information - § 42.8(b)(3) and (4)

Lead Counsel	Anant K. Saraswat, Reg. No. 76,050
Backup Counsel	Elisabeth H. Hunt, Reg. No. 67,336 Daniel D. Smith, Reg. No. 71,278

Service Information	<u>E-mail:</u> ASaraswat-PTAB@wolfgreenfield.com EHunt-PTAB@wolfgreenfield.com DSmith-PTAB@wolfgreenfield.com <u>Post and hand delivery:</u> Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206 <u>Telephone:</u> 617-646-8000 <u>Facsimile:</u> 617-646-8646
------------------------	---

A power of attorney is submitted with the Petition. Counsel for Petitioner consents to service of all documents via electronic mail.

I. INTRODUCTION

Petitioner requests *inter partes* review and cancellation of claims 1-23 of U.S. Patent No. 11,010,336 (“the ’336 patent,” EX1001).

The ’336 patent discloses a “database system” that receives a “user request” via a “dashboard” for “provisioning” a “source database.” EX1001, Abstract, 1:24-28. A “source database” is a database from which “snapshots,” *i.e.* point-in-time records of the database’s state, are taken to create backups. EX1001, 4:46-59, 5:20-23; Bhattacharjee (EX1003), ¶27. “Provisioning” may involve either “creating” a database with user-specified parameters or “associating” an existing database with the database system. EX1001, 3:32-34; Bhattacharjee, ¶31.

The patent describes a “user interface [UI] that solicits information from and conveniently walks the user through the various steps for creating a new database.” EX1001, 4:8-11; Bhattacharjee, ¶47. The information solicited includes parameters such as the database software type (“engine”), database name, and a frequency for taking snapshots. EX1001, 14:14-25, 24:15-25:17, 5:64-67, Figs. 4C, 4E, 4F; Bhattacharjee, ¶¶39-42. The patent’s UI purportedly enables database creation “within minutes,” while database creation in prior-art systems allegedly required “multiple days” of manual work by “multiple entities.” EX1001, 3:32-55, 4:8-19. The patent also “provides the ability to register an existing database with

the database system” to let the user take advantage of “database management services offered by the database system.” EX1001, 4:29-44.

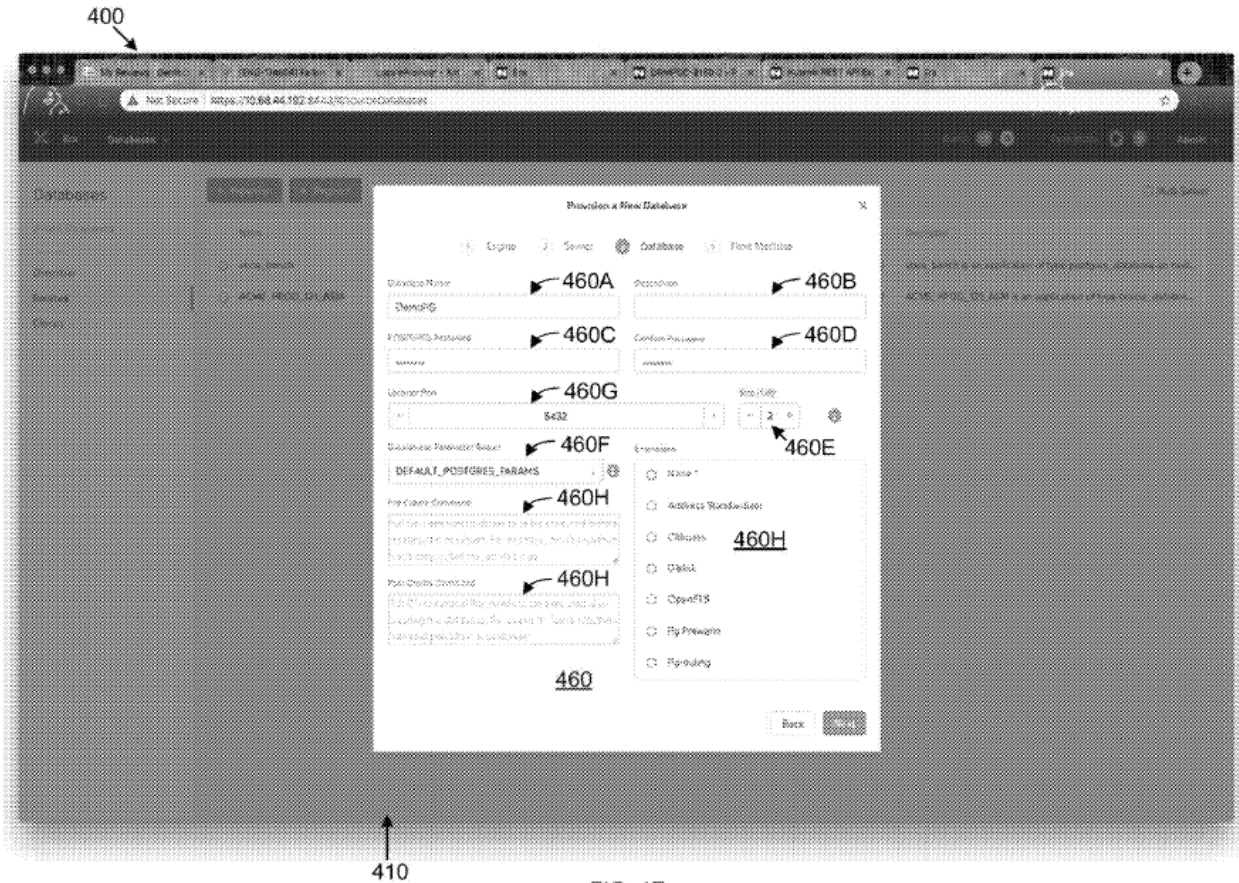


FIG. 4E

EX1001, Fig. 4E¹

Notably, the '336 patent does not purport to disclose any novel database implementation, *e.g.*, any new database software or any new data storage technology. Bhattacharjee, ¶¶52-53. Rather, the patent's claims recite simply

¹ The blurry text is in the '336 patent's original figures.

accepting various parameters via a UI and provisioning a database based on those parameters.

As this Petition demonstrates, all the input parameters recited in the claims were well-known database parameters, as disclosed in, *e.g.*, Sivasubramanian (EX1005) and Zha (EX1006). Nor was there anything novel about receiving the recited parameters as inputs and provisioning a database based on them. As discussed below, years before the claimed 2018 priority date, UIs providing for convenient and quicker parameter input and database provisioning were known.

For example, *in 2009*, Sivasubramanian, assigned to Amazon, disclosed a “control plane” that automates various “workflows” to replace “tedious manual procedures” previously required for tasks such as “provisioning and scaling data storage.” Sivasubramanian, Abstract, 1:46-56, 2:17-21. In contrast to the previously-used manual procedures, which “[took] a significant amount of time and effort,” a user could use a UI to “submit a call including information such as a type of hardware and a version of a database product” to Sivasubramanian’s control plane, which then provisioned the database; thus, Sivasubramanian provided the same functionality *9 years earlier* than the ’336 patent.

Sivasubramanian, 9:49-61, Fig. 5; Bhattacharjee, ¶48-50. Similar database-provisioning UIs were used in commercial products such Amazon’s Relational Database Service and Google’s Cloud SQL. Amazon Relational Database Service

User Guide (“RDS,” EX1013), 13-18, 22; Google (EX1032), 2-3; Bhattacharjee, ¶51.

Web Browser

FILE EDIT VIEW FAVORITES TOOLS HELP

Address

Control interface page

Repository name

Task ☒ Provision ☐ Create ☐ Delete ☐ Modify

Database engine Version:

Storage capacity Units

Partition scheme

Security scheme Version:

Access key Version:

Callout numbers: 502 points to the Modify radio button; 504 points to the Provision radio button; 506 points to the Database engine dropdown; 508 points to the Units dropdown.

Sivasubramanian, Fig. 5

Specify DB Details

Instance Specifications

DB Engine: mysql

License Model: general-public-license

DB Engine Version: 5.6.19a

Review the **Known Issues/Limitations** to learn about potential compatibility issues with specific database versions.

DB Instance Class: - Select One -

Multi-AZ Deployment: - Select One -

Storage Type: - Select One -

Allocated Storage*: 5 GB

Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. [Click here](#) for more details.

Settings

DB Instance Identifier*:

Master Username*:

Master Password*:

Confirm Password*:

* Required

Cancel Previous **Next Step**

RDS, 15

As this Petition demonstrates, Sivasubramanian’s database-provisioning UI was customizable to accept any suitable user-desired parameters, such as by supplementing Sivasubramanian’s disclosed parameters with parameters disclosed in Zha. Furthermore, there was nothing novel about taking snapshots from a database, as this was well-known.

Finally, there was nothing novel about provisioning databases in a “hyperconverged infrastructure system,” as the claims recite. Hyperconverged infrastructures were well-known, and using an interface to provision a database in a hyperconverged infrastructure would have been obvious in view of POSAs’

background knowledge, as explained below in Ground 1. In addition, POSAs would have found it obvious to use an interface to provision a database in the specific example of a hyperconverged infrastructure taught by Shekar (EX1007), as explained below in Ground 2.

All the '336 patent's claims would have been obvious as demonstrated below.

II. STANDING CERTIFICATION

Petitioner certifies that the '336 patent is available for *inter partes* review and that Petitioner is not barred or estopped from requesting *inter partes* review of the challenged claims. 37 C.F.R. § 42.104(a).

III. GROUNDS FOR UNPATENTABILITY

The table below identifies the references, applicable claims, and basis for each unpatentability ground. Sivasubramanian, Zha, and Shekar are prior art under post-AIA §§102(a)(1)-(2). Zha and a descendant of Sivasubramanian were of record during prosecution, but neither was substantively discussed or applied in any rejection, nor were the Sivasubramanian+Zha or Sivasubramanian+Shekar+Zha combinations ever considered. *Infra* §VIII.B.

Ground Number and Reference(s)		Claims	Basis
1	Sivasubramanian, Zha	1, 5-8, 10-15, 19-21	§103
2	Sivasubramanian, Shekar, Zha	1-23	§103

IV. THE '336 PATENT

A. Described Embodiments

The '336 patent discloses a “database system.” EX1001, Abstract.

Databases (which long predated the patent) are collections of organized data.

Bhattacharjee, ¶¶27-28; Quernemoen (EX1008), 1:36-37; Heritage (EX1009), 462.

Databases can be created and managed by different types of database software, *e.g.*

“Oracle” and “MySQL,” which the patent refers to as “database engine types.”

Bhattacharjee, ¶29; EX1001, 14:14-25; Sivasubramanian, 2:44-47.

The '336 patent’s “database system” includes a “database provisioning system” that provides “database provisioning services.” EX1001, 10:44-57; Bhattacharjee, ¶30. The patent states that “[d]atabase provisioning services involve creating and/or associating databases with the database system for management and use”—*i.e.*, the patent’s “provisioning system” can either “create” a new database and provide resources to it, or provide resources to an already-existing database. EX1001, 3:32-34, 15:62-66; Bhattacharjee, ¶¶31-36; EX1010, 2:46-50; EX1011, [0004]. Accordingly, the patent describes two different “example user interfaces”— for “creating” and “registering,” respectively, a “database.” EX1001, 22:23-25.

“[T]o initiate a database provisioning service,” the user navigates to a “database page” having a “create button” and a “register button.” EX1001, 23:43-

24:3, Fig. 4B (annotated below); Bhattacharjee, ¶¶37-38. The exemplary database-creation interface is shown in Figs. 4C-4G. EX1001, 22:25-26.

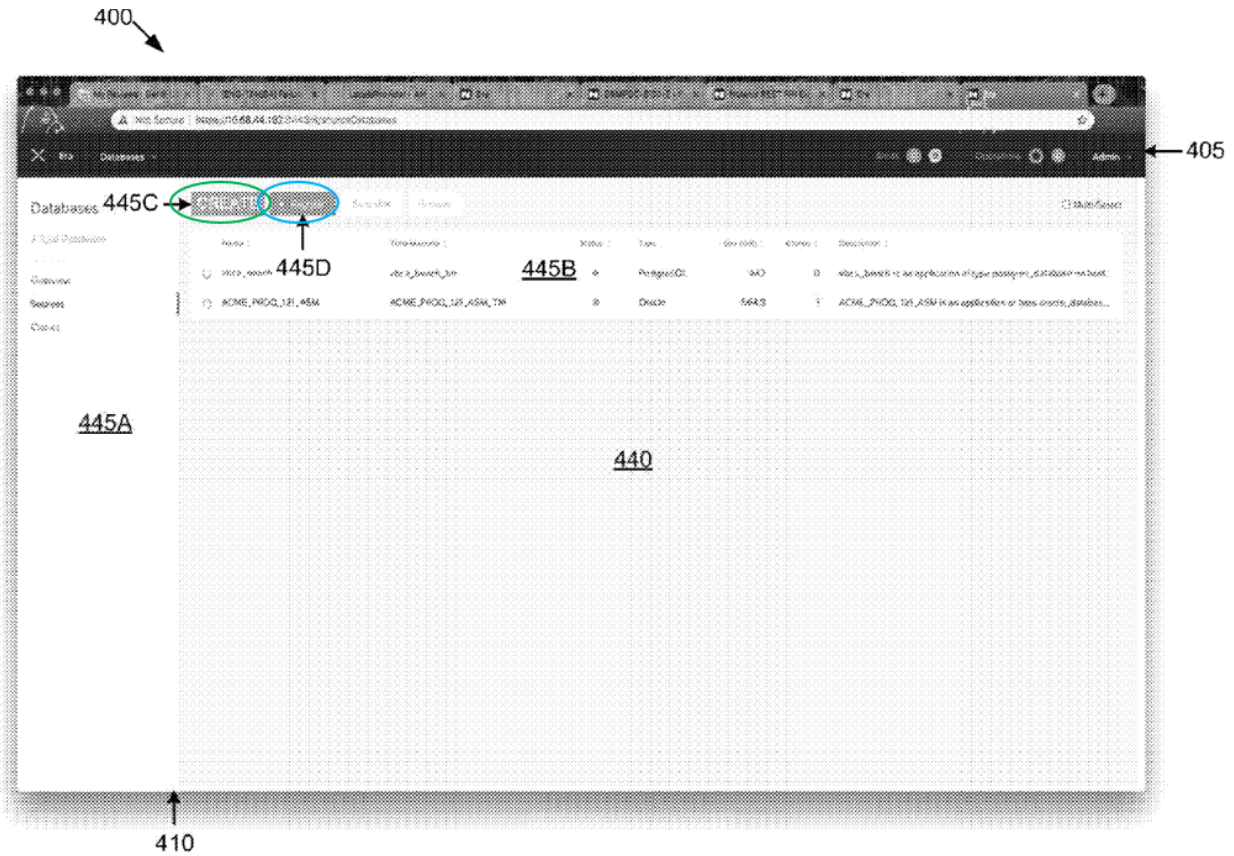
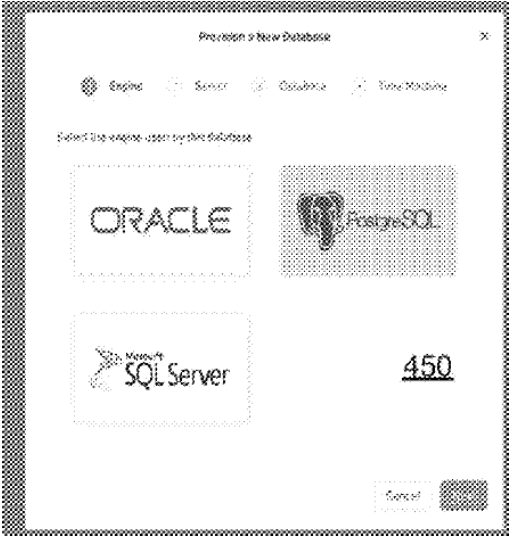
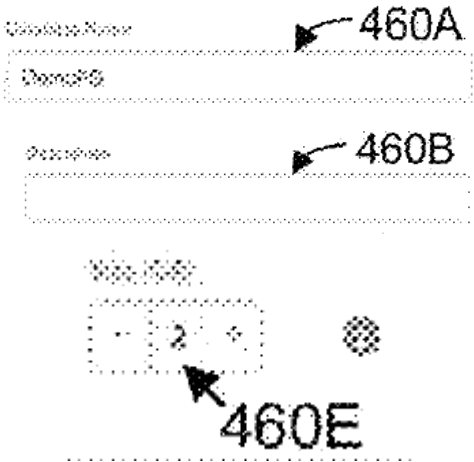
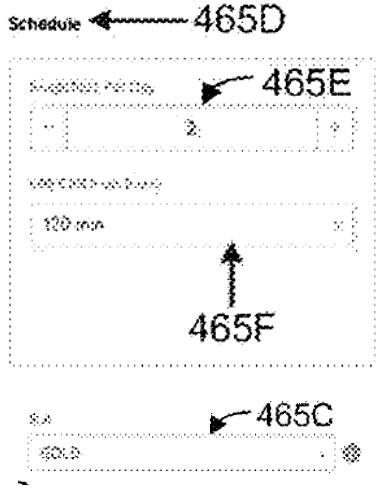


FIG. 4B

EX1001, Fig. 4B (annotated)

After pressing “create,” the user can enter various parameters on subsequent screens. EX1001, 24:4-10; Bhattacharjee, ¶39. Examples are shown below with excerpts from relevant figures.

'336 Specification	'336 Figures
“database engine type,” <i>e.g.</i>, “Oracle, PostgreSQL, etc.” (EX1001, 14:22-24, 24:15-23, FIG. 4C);	 FIG. 4C (excerpt)
“name 460A,” “description 460B,” “size 460E,” (EX1001, 24:49-63, Fig. 4E);	 FIG. 4E (excerpts)

'336 Specification	'336 Figures
“protection schedule 465D” and “SLA [service level agreement] level 465C” (EX1001, 25:1-17, Fig. 4F)	 FIG. 4F (excerpts)

Once the parameters are entered, the user presses a button to send the parameters to a “database creation system” to “start the database creation.” EX1001, 25:17-22; Bhattacharjee, ¶40.

The user invokes UIs for “registering an existing database” by pressing “register” instead of “create” on the “database page.” EX1001, 25:43-54. The user can then enter various parameters on subsequent screens, including an “SLA and protection schedule.” EX1001, 25:60-67. “Upon receiving all of the parameters from the user, the database registration system... registers the database.” EX1001, 26:1-2; Bhattacharjee, ¶¶41-45.

Although the exemplary interfaces discussed above are graphical, the '336 patent notes that its “dashboard” “receive[s] user inputs” via either a “graphical

user interface (‘GUI’)” or a “text-based user interface[.]” EX1001, 11:61-64, 12:28-43; Bhattacharjee, ¶46.

B. Prosecution History

The examiner rejected the originally-filed claims over Weissman (EX1018) in view of EX1019. EX1002, 256. Applicants scheduled an examiner interview and, in the agenda, (1) proposed amending claim 1 to recite what became Element [1F] and (2) argued Weissman did not disclose as-filed claim 7 because Weissman did not disclose a “first frequency” for capturing snapshots and a “second frequency” for capturing transactional logs. EX1002, 275-276. The examiner agreed that the incorporating claim 7 into claim 1 overcame the prior art rejection. EX1002, 278. The applicants then amended independent claims 1, 12, and 19 to add Elements [1F-G]/[12F-G]/[19F-G], and also amended various dependent claims. EX1002, 287-291; *infra* §X. The examiner then allowed the claims; in the statement of “[r]easons for allowance,” the examiner said “the prior art does not teach or fairly suggest” the subject matter of the independent claims, but did not identify or discuss any specific reference. EX1002, 324-326; Bhattacharjee, ¶¶54-55.

C. Person of Ordinary Skill in the Art (“POSA”)

The ’336 patent relates to UIs for deploying database services. EX1001, Abstract, 1:24-2:9, 3:45-49; *supra* §IV.A; Bhattacharjee, ¶¶56-57. A POSA at the

time of the claimed December 27, 2018 priority date would have had a bachelor's degree in computer science, computer engineering, or a related field, with two years of academic and/or industry experience with database technologies. A higher level of education may substitute for less experience. Bhattacharjee, ¶¶58-60.

V. CLAIM INTERPRETATION

Claim terms are construed herein using the standard used in civil actions under 35 U.S.C. § 282(b), in accordance with the ordinary and customary meaning as understood by a POSA and the prosecution history. 37 C.F.R. § 42.100(b).

VI. GROUND 1: CLAIMS 1, 5-8, 10-15, 19-21 ARE UNPATENTABLE OVER SIVASUBRAMANIAN+ZHA

A. Sivasubramanian (EX1005)

1. Overview

Sivasubramanian concerns “the *creation, provisioning*, and management of data stores and instances” using a “control environment.” Sivasubramanian, Abstract, Fig. 3, 10:5-10; Bhattacharjee, ¶¶61-64. “[D]ata store’ refers to... any device or combination of devices capable of storing, accessing, and retrieving data”; such a device or devices may include “databases.” Sivasubramanian, 3:15-19.

Sivasubramanian’s “control environment” (a/k/a “control plane”) manages “workflows” that perform tasks in a “data plane” (a/k/a “data environment”); a

“workflow” is a “sequence of tasks” for “specific job.” Sivasubramanian, Abstract, 2:21-24, 6:25-26, Fig. 2; Bhattacharjee, ¶¶65-66. “Workflows” may be initiated in response to requests submitted to the control plane using graphical or textual interfaces. Sivasubramanian, 2:26-32, 5:4-10, 5:35-44, 6:12-23; *infra* §VI.A.1.a; Bhattacharjee, ¶¶67-71.

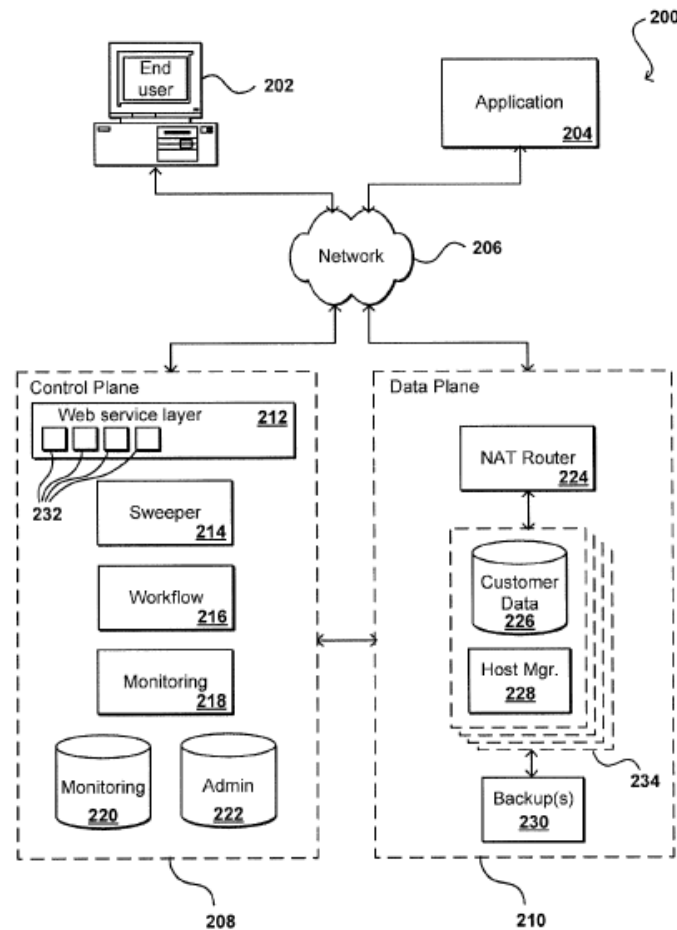


FIG. 2

Sivasubramanian, Fig. 2

One “workflow” the control plane can manage is “create database.” Sivasubramanian, 6:59-65. As explained below, POSAs understood that Sivasubramanian’s “create database” workflow includes what the ’336 patent calls “provisioning” a database, for multiple reasons. Bhattacharjee, ¶¶72.

First, Sivasubramanian says the “create database” workflow “includes tasks such as *provisioning a data store instance*.” Sivasubramanian, 6:59-65. Sivasubramanian uses “data instance” or “data store instance” to refer to a database together with the resources—such as processing power, memory, and storage—that the database uses. Sivasubramanian, 6:59-65, 11:48-51, 9:64-10:4; *see also* Sivasubramanian, 6:43-50 (workflow template for creating “*database*” based on user choice of “MySQL” v. “Oracle” “*instance*”); Bhattacharjee, ¶¶73.

Second, Sivasubramanian also references using a “Create*Database*” API (“application programming interface”) to “*provision*” a “*repository*,” and then using “Describe*Database*” to check if the “repository” is “provisioned.” 11:47-56, 2:26-32; Bhattacharjee, ¶¶74. POSAs understood that the “repository” is a *database* from the referenced API names “Create*Database*” and “Describe*Database*” and because Sivasubramanian describes a “database” as a type of “repository” and frequently uses those terms interchangeably. Sivasubramanian, 15:37-38 (“*repository* such as a relational *database*”), 4:58-62. 4:49, 5:15-20, 1:47-56, 11:52-56, 12:24-33, 13:18-21, 12:64-13:5, 13:27-33, Fig. 5; Bhattacharjee, ¶¶75-

77. In Sivasubramanian, “workflows” may be initiated in response to “requests” submitted via “APIs”; thus, POSAs would have understood from the name that the “Create**Database**” API initiates the “create database” workflow. Sivasubramanian, 2:26-32, 5:4-10, 5:35-44, 6:12-23; Bhattacharjee, ¶78.

Third, the ’336 patent uses “provisioning” to encompass creating a database and giving it resources. *Supra* §IVVI.A. As noted above, Sivasubramanian discloses the “create database” workflow “provisioning” a “repository,” *i.e.* database, and also provisioning a “data instance” representing a database and its resources. POSAs understood from the name “create database” that the workflow “**creates**” a database and then gives it resources, *e.g.* the resources in the data instance; thus, Sivasubramanian’s “create database” workflow does the same thing the ’336 patent describes as “provisioning.” Bhattacharjee, ¶79.

Fourth, the ’336 patent says “[d]atabase provisioning services involve creating and/or associating databases with the database system **for management and use.**” EX1001, 3:32-34. Likewise, following completion of Sivasubramanian’s “create database” workflow that includes “provisioning,” the database, and the “data instance” on which it runs (as discussed above), can be **managed** by the control plane, and also can be **used**. *E.g.*, Sivasubramanian, 6:59-67 (“‘create database’ workflow... might include[]” providing resources that “the customer can use to access... the data instance”), 9:4-8 (similar), 2:26-44 (similar),

9:45-65 (similar), 12:1-36 (describing API for modifying database), 13:18-15:31 (describing management actions available after creation), , 19:32-64 (discussing workflow for backup); Bhattacharjee, ¶80.

Additionally, as noted *supra* §IV.A, the '336 patent also uses the term “provisioning” to refer to “registering” an existing database and providing resources to it. EX1001, 3:32-34. The patent describes an exemplary “registering” interface where a user identifies an existing database and then enters parameters for providing it resources. EX1001, 25:43-67, Fig. 5A-5E; Bhattacharjee, ¶81. As discussed *infra* §VI.A.1.a, Sivasubramanian likewise discloses UIs that let users identify an existing database and specify parameters for providing it resources.

a. Database-Provisioning UIs

Sivasubramanian says “F[igure] 5 illustrates an example of a display” that is an “example” of an “interface page... allow[ing] a user to ***submit requests*** to the ***control plane***.” Sivasubramanian, 12:49-55. Sivasubramanian also describes Figure 5’s exemplary “display” and the below-referenced command-line tool as alternative ways to submit “calls to the control plane.” Sivasubramanian, 12:53-13:5. Thus, POSAs understood Sivasubramanian to disclose using a GUI like Figure 5’s to submit requests to invoke workflows. *Infra* §VI.A.1.a.i; Bhattacharjee, ¶82.

Sivasubramanian also discloses a “command line tool” that is an “example” way to “submit Web services calls to the control plane.” Sivasubramanian, 12:62-66. Sivasubramanian also says users can “call a Web service” to invoke the “‘CreateDatabase’ or similar API”; this API invokes the “create database” workflow (*see supra* §VI.A.1). Sivasubramanian, 11:47-52. Thus, POSAs understood that the command-line tool could be used to submit requests to invoke workflows. *Infra* §VI.A.1.a.ii; Bhattacharjee, ¶83.

As explained *infra* §§VI.A.1.a.i-VI.A.1.a.ii, the GUI and command-line, respectively, can be used to invoke workflows to “provision” a database (in the ’336 patent’s meaning) by creating a new database and allocating it resources and/or allocating resources to an existing database. Throughout this Petition, the term “database-provisioning UI” refers to a UI that performs either aforementioned “provisioning” process. Bhattacharjee, ¶84.

i. GUI

Sivasubramanian’s Figure 5 “display” lets the user “select an action to be performed 504,” and select “options for the actions” including “database engine” (506) and “version information” (508). Sivasubramanian, 12:53-62; Bhattacharjee, ¶85.

Web Browser

FILE EDIT VIEW FAVORITES TOOLS HELP

Address

Control interface page

Repository name: Repository A

Task: ☒ Provision ☐ Create ☐ Delete ☐ Modify

Database engine: MySQL Version: 5

Storage capacity: 150 Units: GB

Partition scheme:

Security scheme: Version: 0

Access key: Version: 0

Save Cancel

Sivasubramanian, Fig. 5

In Figure 5 (annotated above), the “action to be performed 504” (Sivasubramanian, 12:59-60), labeled “**Task**,” shows the “**Provision**” option selected. POSAs understood that the “Provision” “action”/“Task” provisions a *database*, and that the “Repository name” is a database name, because the interface includes an option for selecting a “**Database** engine,” and because of Sivasubramanian’s repeated use of “repository” for “database” (noted *supra* §VI.A.1). Sivasubramanian, 12:56-62, Fig. 5; Bhattacharjee, ¶86. Furthermore, the figure shows that the “Repository [*i.e.*, database] name” field lets the user select from options via a drop-down menu (indicated by the down arrow),

indicating that the “Provision” “action”/“Task” provisions an already-*existing* database the user selects, rather than creating a new database. Bhattacharjee, ¶86; *see also* Sivasubramanian, 12:38-40.

As shown above, Figure 5 also includes a “Create” option. POSAs understood that the “Create” option presents an interface for creating a *database*, at least because (1) Sivasubramanian describes the GUI and the “rds-create-*database*” command-line command as alternative methods for creating a database, and (2) Sivasubramanian describes “*database* engine” as an “option” for the “action 504,” labeled “Task 504” in Figure 5 (annotated above). Sivasubramanian, 12:49-13:5; Bhattacharjee, ¶87.

Sivasubramanian says Figure 5’s interface “include[s] options that allow the user to enter information needed to perform a control action on the data plane.” Sivasubramanian, 12:55-57; Bhattacharjee, ¶88. Thus, POSAs would have understood Sivasubramanian to teach, or at minimum render obvious, that the fields for database-creation parameters presented in the “Create” interface would include fields for the parameters shown in Figure 5 for the “Provision” interface, as explained below.

Sivasubramanian’s create-database workflow includes provisioning the database by allocating it resources. *Supra* §VI.A.1. Thus, POSAs understood that the “Create” interface likewise included the fields on the “Provision” interface.

Bhattacharjee, ¶89. Further, Sivasubramanian discloses an “rds-create-database” command-line command for creating a database. Sivasubramanian, 13:1-5; *supra* §VI.A.1.a.ii. The parameters Sivasubramanian shows “rds-create-database” accepting include parameters shown in Figure 5’s “Provision” task screen, including name, engine, and size. Sivasubramanian, 13:1-5 (“rds-create-database” command accepting “dbname,” “size,” “engine”), Fig. 5 (showing fields for “Repository name,” Database engine,” and “Storage capacity”); Bhattacharjee, ¶90 (explaining POSAs understood “db” in “dbname” to be a common abbreviation for “database,” and understood “capacity” in GUI to be equivalent to “size” in “rds-create-database” command) (citing Sivasubramanian, 12:42-44).

Additionally, POSAs understood, or at minimum would have found obvious, that in the “Create” interface, the user could enter a new “Repository name,” *i.e.* database name as discussed above, for the new database being created, rather than selecting the name of a previously-created database to provision as in the “Provision” interface. Bhattacharjee, ¶91.

To the extent not taught by Sivasubramanian, POSAs would have found implementing the “Create” task GUI to accept inputs of parameters shown on the “Provision” task GUI to be obvious as a well-known UI implementation that beneficially increased user convenience by letting the “Create” UI accept parameters needed to provision a created database (functionality Sivasubramanian

itself teaches, as discussed above). Bhattacharjee, ¶92; RDS, 13-18 (describing “Creating a MySQL DB Instance” UI that creates a “provisioned” database).

Additionally, POSAs would have understood that accepting a new database name was logical in a UI for creating a new database. Bhattacharjee, ¶92. POSAs would have reasonably expected success implementing the “Create” UI in this manner because it was within POSAs’ skill. *Infra* §VI.A.2.

As discussed *supra* §VI.A.1.a, Sivasubramanian’s Figure 5 GUI can be used to submit requests to the control plane to initiate workflows. Thus, POSAs understood, or at least would have found obvious, that the “Create” option in the Figure 5 GUI invokes the “create database” workflow, which *provisions* a new database (under the ’336 patent’s meaning) as discussed *supra* §VI.A.1. POSAs also understood, or at least would have found obvious, that the “Provision” task likewise invokes a workflow for provisioning an existing database. Sivasubramanian, 5:16-20, 7:24-28; Bhattacharjee, ¶93.

ii. Command-Line

The command-line tool presents a text-based interface (command line) for the user to enter text in a specified command format to input parameters for a task such as database creation. Sivasubramanian, 12:64-13:5. The command-line format accepts input parameters as a list of pairs of parameter names and values, as shown in the example “rds-create-database” command:

```
rds-create-database --identifier customerprod --dbname customer --  
size 80 [-]-class small --engine mysql5.1 --master master_username --  
password master_password -port 4030
```

Sivasubramanian, 12:64-13:5; Bhattacharjee, ¶94.

Sivasubramanian teaches that the “rds-create-*database*” command creates a database, as indicated by the command’s name and by Sivasubramanian’s description of it as an example command for “generat[ing] a request to create a new data *repository*,” a term which in Sivasubramanian encompasses a “database,” as discussed *supra* §VI.A.1. Sivasubramanian, 12:64-13:5; Bhattacharjee, ¶95.

The command-line tool can be used to submit requests to the control plane to initiate workflows. *Supra* §VI.A.1.a. Thus, Sivasubramanian teaches, or at least renders obvious, that the “rds-*create-database*” command invokes the “create database” workflow, which *provisions* a new database (under the ’336 patent’s meaning) as discussed *supra* §VI.A.1. Bhattacharjee, ¶96.

Furthermore, POSAs would have understood the “rds-create-database” command as just one representative example discussed for illustration purposes, and would have been motivated to similarly implement a command-line version of the above-discussed “Provision” task in Figure 5’s GUI that provisions an existing database, as this would beneficially provide parallel functionality with the GUI. POSAs would have expected success because implementing such a command,

which may have been called, *e.g.*, “rds-provision-database,” was within a POSA’s skill as discussed *infra* §VI.A.2. Bhattacharjee, ¶97.

b. Backup “Workflow”

Sivasubramanian discloses a “recovery and backup” “workflow[.]”

Sivasubramanian, 6:31-42; Bhattacharjee, ¶98. Sivasubramanian says the control plane can use “software frameworks” for “database backup and recovery,” and also says a “framework” can handle “backing up both the data files and any associated log files.” Sivasubramanian, 16:21-27, 19:25-28. Sivasubramanian describes one exemplary “approach for backing up data” whereby a “su[s]pendDatabaseForBackup” command is used to “suspend[] data operations” while “s[n]apshots are taken of the appropriate data volumes.” Sivasubramanian, 19:32-41; Bhattacharjee, ¶¶99-100.

Additionally, in Sivasubramanian’s above-mentioned “approach for backing up data,” “log files” are “copied” while the database is suspended.

Sivasubramanian, 19:32-41. The log files are used for “replay[ing]... *transactions* in case the data files have to be *restored*”; thus, Sivasubramanian’s “log files” are what other art and the ’336 patent refer to as “transaction logs” or “transactional logs.” Sivasubramanian, 19:65-20:1; Zha, 10:6-11; RDS, 112; EX1001, 36:49-59; Priebe (EX1060), 1:14-22, 1:61-63; EX1061, 4:17-30; Bhattacharjee, ¶101.

2. Motivation to Customize Sivasubramanian's UI

Although Sivasubramanian's Figure 5 shows a particular example set of configuration parameters that can be input, POSAs understood that the UI shown could beneficially accept additional inputs not expressly shown. Bhattacharjee, ¶102. Sivasubramanian says Figure 5 is an “*example* of a display” in an “interface” including “options that allow the user to enter *information needed* to perform a control action on the data plane,” and notes that “*other* such values” aside from those shown can be specified. Sivasubramanian, 12:53-64; Bhattacharjee, ¶103. Sivasubramanian also teaches that the user interface to the control plane is flexible and enables the user to enter whatever options are appropriate to accomplish a desired workflow. Sivasubramanian, 11:49-52 (“The user can call a Web service to specify *any* desired values for an instance type..., storage size, repository name, port, *and other such values.*”)², 2:23-26, 11:41-47, 6:7-12, 6:45-52; Bhattacharjee, ¶104. Furthermore, Sivasubramanian's discussion of the “rds-create-database” command shows different options for parameters than those shown in Figure 5, indicating that other parameters could similarly be added to Figure 5's GUI. Sivasubramanian, 13:1-5 (“rds-create-database” command including parameters for “identifier,” “class,” “password”); Bhattacharjee, ¶105.

² Emphases added unless otherwise indicated.

Such UI implementations were well within a POSA's skill. Bhattacharjee, ¶106. Sivasubramanian describes Figure 5 as an example page rendered in a "browser application." Sivasubramanian, 12:52-55. Sivasubramanian also discloses implementing its functionality on "[w]eb server[s]" running "applications" implemented in "any programming language." Sivasubramanian, 23:4-16. POSAs had the skill to use "programming language[s]" to implement an application to display a version of Figure 5's exemplary UI including additional options. Bhattacharjee, ¶106.

POSAs would have been motivated for the same reasons to implement Sivasubramanian's command-line tool to accept additional inputs not shown in the example at 13:1-5; furthermore, it would have been within a POSA's skill to use a "programming language" to implement such a command line, and to implement additional commands such as the "rds-provision-database" command discussed *supra* §VI.A.1.a.ii. Sivasubramanian, 23:4-16; Bhattacharjee, ¶107.

B. Zha (EX1006)

Zha teaches techniques for creating a "virtual database"—*i.e.*, a copy of an existing database from a particular point in time—used for purposes including "backup." Zha, Abstract; 6:49-59, 24:24-27; Bhattacharjee, ¶¶108-109. Zha's "[m]ethods of creating a virtual database (VDB) read different point-in-time copies of a source database," which Zha also refers to as a "production database." Zha,

1:66-2:6. Zha also teaches retrieving “transaction logs” from the source/production database; these logs can be combined with point-in-time copies to “reconstruct a copy of a database.” Zha, 9:63-10:11, 15:58-62, 18:1-3; Bhattacharjee, ¶¶110-111.

Zha’s “database storage system” (where database copies are stored) can be configured by a “database administrator” using a “system configuration manager.” Zha, 6:11-58, 11:26-28; Bhattacharjee, ¶¶112-113. This “manager” lets a user “setup... the *schedule*” specifying the “frequency and times during the day” for “retriev[ing] *PIT [point-in-time] copies of databases*” and “retriev[ing] updates to online *transaction logs*.” Zha, 11:33-44; Bhattacharjee, ¶113.

POSAs understood that Zha’s “point-in-time copies” are “snapshots” of the source database at the time the copy was made, for multiple reasons. The term “snapshot” generally referred to a record of the database’s state at a point in time. Bhattacharjee, ¶114; Simek (EX1014), [0002]; Ohno (EX1015), 1:54-61.

Furthermore, Zha states that “transaction logs can be applied to a VDB [virtual database] created based on a PIT copy... to create a *snapshot* of the source database for a time point that occurs *after* the PIT copy was made.” Zha, 17:63-66. Because applying transaction logs creates a “snapshot” of the database at a time *after* the PIT copy was made, POSAs understood that the PIT copy itself represents a “snapshot” *at* the time it was made, since Zha also explains that the

transaction logs “can be used to reconstruct a *copy* of a database... corresponding to times in the past in *between* the times as which *point-in-time copies* are made.” Zha, 10:6-11; Bhattacharjee, ¶115.

Zha also discloses “policies defined by a database administrator” for operations including “purging of information... including point-in-time copies of databases and transaction log information.” Zha, 11:45-54. Such “policies” include a “schedule for purging the point-in-time copy after a retention period” (Zha, claim 11, 35:18-20) and “a schedule for purging the transaction logs after a retention period” (Zha, claim 13, 35:24-26). Bhattacharjee, ¶116.

C. Sivasubramanian+Zha Combination

POSAs seeking to implement Sivasubramanian’s “backup” workflow discussed *supra* §VI.A.1.b would have considered references like Zha regarding certain implementation details Sivasubramanian is silent/agnostic about. Although Sivasubramanian and Zha use slightly different terminology, both discuss database backup using snapshots and transaction logs, with Zha’s teachings supplying further details; thus, POSAs would have found it obvious to implement Sivasubramanian using Zha’s teachings in multiple independent ways, as discussed below. Bhattacharjee, ¶117.

Sivasubramanian’s “snapshots” discussion appears in a paragraph describing “[an] approach for backing up data” that immediately follows a paragraph

introducing Sivasubramanian’s discussion of “*repository* and data backup” and also noting that “*approaches* can differ from various *database engines*.”

Sivasubramanian, 19:20-35. Sivasubramanian’s language and structure thus confirm that Sivasubramanian discloses *database* snapshots. Bhattacharjee, ¶118; *see also* Sivasubramanian, 15:1-15 (earlier discussion of “tak[ing] a snapshot of a production instance” so that a “test instance” created with a “create-*database*” command can be “fully populated and comparable with the production [data]”).³

Additionally, Sivasubramanian’s exemplary “approach” uses a “su[s]pend*Database*ForBackup” command to “suspend[] data operations” while “s[n]apshots are taken of the appropriate data volumes.” Sivasubramanian, 19:32-41; Bhattacharjee, ¶119. Sivasubramanian also discloses storing a database’s data on “volume[s]” allocated to a database’s storage “instance.” Sivasubramanian, 6:59-63, 8:22-25, 15:56-60. Moreover, POSAs understood that taking snapshots of underlying data volumes was a typical and customary way to take a database snapshot. Bhattacharjee, ¶120; EX1022, 1; Ohno, 1:28-29, 1:54-2:29; EX1055, 17:23-40. Thus, POSAs understood Sivasubramanian to disclose taking *database*

³ Sivasubramanian 15:3 says “production date,” but POSAs understood this is a typo. Bhattacharjee, ¶118.

snapshots by taking snapshots of the underlying data volumes. Bhattacharjee, ¶121.

POSAs understood that “tak[ing]” “snapshots” of a database in Sivasubramanian (Sivasubramanian, 15:1-5, 19:20-64) is analogous to what Zha refers to as “retriev[ing] PIT [point-in-time] copies” (Zha, 11:33-44, 17:63-66), and likewise understood that “backing up” the “log files” in Sivasubramanian (Sivasubramanian, 19:65-20:11) is analogous to what Zha refers to as “retriev[ing] updates to online transaction logs” (Zha, 11:33-44); both references’ processes result in the snapshot/log being stored in some storage location used for storing snapshots/logs. *Compare* Sivasubramanian, 19:32-35 *with* Zha, 2:19-24, 9:67-10:3; Bhattacharjee, ¶122.

Sivasubramanian does not restrict implementation details such as daily frequency of retrieving snapshots and transaction logs, what time of day these events occur, or retention duration of snapshots and logs. Sivasubramanian, 19:20-20:11; Bhattacharjee, ¶123. However, POSAs understood it was typical to collect snapshots, logs, and other database-backup information at user-configurable frequencies and times and retain such information for user-configurable durations. Bhattacharjee, ¶123; Bhasin (EX1016), [0009]; RDS, 16, 111-12; Tekade (EX1017), [0046], Fig. 6; Simek, [0034], Fig. 2. Furthermore, POSAs understood that taking and storing snapshots and transaction logs required computing

resources and storage space and potentially impacted the database’s performance; thus, POSAs understood that allowing users of Sivasubramanian’s control plane to specify snapshot and log frequencies, times, and retention periods beneficially enabled users to tune those parameters to suit their needs. Bhattacharjee, ¶124; *see, e.g.*, Madany (EX1023), [0023]; RDS, 5, Sivasubramanian, 19:31-41; Ohno, 1:54-61.

Therefore, POSAs seeking to implement Sivasubramanian’s exemplary “approach for backing up data” (Sivasubramanian, 19:32) would have been motivated to use an administrator-defined “schedule” including a “frequency” and “times during the day” for taking snapshots and backing up transactional/transaction logs, and to store them for administrator-defined “retention periods,” as Zha discloses. Zha, 11:33-54, 35:18-20, 35:24-26; Sivasubramanian, 19:20-20:11; Bhattacharjee, ¶125.

In the resulting “Sivasubramanian+Zha” combination, Sivasubramanian’s database-provisioning UIs discussed *supra* §VI.A.1.a accept user-defined snapshot and transaction-log collection frequencies, times during the day, and/or retention periods, as detailed *infra* §§VI.C.1-VI.C.2. Additionally, in the combination, the provisioning process would setup the database to have snapshots and logs (1) collected at the respective user-defined frequencies and times and (2) stored for the respective user-defined periods. Sivasubramanian, 19:32-39; Bhattacharjee, ¶126.

Alternatively, POSAs would have found it obvious to employ Zha’s technique of creating VDBs (alternatively or additionally to creating Sivasubramanian’s snapshots) to beneficially achieve the “efficient[] use [of] storage and computing resources” Zha’s technique provides. Zha, 1:63-65; Bhattacharjee, ¶127. POSAs would have been motivated to customize Sivasubramanian’s database-provisioning UIs to let the user set snapshot and log schedules and retention periods for the same above-discussed reasons, and the resulting system would incorporate the UI elements detailed *infra* §§VI.C.1-VI.C.2 into Sivasubramanian’s database-provisioning UIs. The resulting system would also setup the provisioned database to have “point-in-time copies,” *i.e.* snapshots, and “transaction logs” “retriev[ed]” from it, as Zha teaches, based on user-entered parameters as discussed above. Zha, 11:33-44; *supra* §VI.B; Bhattacharjee, ¶127.

POSAs would reasonably have expected success with either alternative above-discussed implementation of Sivasubramanian+Zha. As detailed *infra* §§VI.C.1-VI.C.2, POSAs had the skill to implement Sivasubramanian’s database-provisioning UIs to accept the inputs Zha teaches. POSAs also had the skill to have the provisioning process setup the database to have snapshots and logs taken/retrieved and retained based on user-entered parameters, since Sivasubramanian itself discloses the control plane executing tasks based on user-entered parameters. *E.g.*, Sivasubramanian, 11:41-52, 12:64-13:5, 19:32-39;

Bhattacharjee, ¶¶128-130. Moreover, Sivasubramanian says the “approach” it describes for backup is “[a]n” approach, meaning that Sivasubramanian’s control plane could accommodate other approaches like Zha’s. Sivasubramanian, 19:32; Bhattacharjee, ¶131. POSAs had the skill to implement Sivasubramanian’s control plane to use Zha’s “virtual database” technique; Zha’s teachings can be implemented in software, and Sivasubramanian likewise discloses implementing the control plane using software “framework[s]” that can perform tasks including “backup and recovery.” Bhattacharjee, ¶132; Zha, 32:49-52; Sivasubramanian, 16:21-28.

1. UI for retention period

It was well-known to specify a “service level” setting retention periods for database-protection-related data such as snapshots and transaction logs; such levels were often specified in a “service level agreement” (“SLA”). Bhattacharjee, ¶133; Tekade, [0046], Fig. 6 (“SLA protecting a database” including “snapshot policy” specifying retention period); EX1024, [0120]; EX1025, [0004]; EX1026, 11:39-43; EX1027, 19:32-35. It was also well-known background knowledge to define multiple SLA “levels” named, *e.g.*, “gold,” “silver,” etc. Bhattacharjee, ¶134; Gomatam (EX1028), [0046], [0056]-[0058]; Higginson (EX1029), 9:4-7; EX1030, 3:62-4:6.

In Sivasubramanian+Zha, POSAs would have been motivated to have control-plane users specify Zha's configurable snapshot/log retention periods (*supra* §VI.B) by specifying an SLA level, for multiple reasons. POSAs understood that it was beneficial to give users familiar UIs, and that users were familiar with SLAs to specify retention periods, as discussed above (and also familiar generally with specifying time intervals). Bhattacharjee, ¶135, Tekade, Fig. 6; RDS, 15-16; Simek, Fig. 2. Furthermore, POSAs understood that letting users choose from predefined sets of parameter values was well-known and increased user convenience. Bhattacharjee, ¶136; Higginson, 9:49-10:5; EX1031, 1:16-29. Thus, in either above-discussed Sivasubramanian+Zha implementation, POSAs would be motivated to let the user specify transaction log and snapshot retention periods in the database-provisioning UIs discussed *supra* VI.A.1.a via the well-known technique of SLA level selection, to beneficially increase user familiarity and convenience. Bhattacharjee, ¶137.

Furthermore, POSAs understood that accepting input of the SLA in connection with provisioning new or existing databases beneficially allowed a database to be protected as soon as it is provisioned, since parameters for snapshot and log capturing and retention would already be specified. Bhattacharjee, ¶138. Moreover, it was well-known in prior-art database-provisioning UIs both to (1) accept inputs related to retention of snapshots or other backup-related data; and (2)

accept SLA-level selection. Bhattacharjee, ¶¶139-141; Gomatam, [0046], [0056]-[0058]; Higginson, 9:49-10:12, Fig. 4; RDS, 16-17; Oracle, 7; Google, 2-3.

In Sivasubramanian’s GUI, POSAs would have been motivated to implement SLA selection via a selectable interface like a drop-down menu, since such interfaces were well-known, including in Sivasubramanian. Bhattacharjee, ¶142; Sivasubramanian, 15:59-60, Fig. 5 (*e.g.*, “Database engine”); Tekade, Fig. 6; RDS, 15. And POSAs would have been motivated to implement the command-line provisioning commands (*see supra* §VI.A.1.a.ii) to accept a parameter whose value represents a level, like the “engine” parameter that specifies an engine. Sivasubramanian, 13:1-5. POSAs would have reasonably expected success with such UI implementations, because SLAs were well-known (*see above*), and because customizing Sivasubramanian’s UIs was within a POSA’s skill (*supra* §VI.A.2). Bhattacharjee, ¶142

Furthermore, POSAs would have found it obvious to have the SLA specify either one retention period for both snapshots and logs or separate periods for each. POSAs understood there were two options—uniform or non-uniform—and either was an obvious design choice. POSAs would have additionally recognized that uniform periods beneficially simplify snapshot/log management, and POSAs would have reasonably expected success because it was simple and well-known to

apply one duration to all snapshots/logs. Bhattacharjee, ¶143; Tekade, [0046]; Bhasin, [0009]; RDS, 5, 111-12.

2. UI for protection schedule

As explained below, POSAs would have been motivated to customize Sivasubramanian's Figure 5 GUI to accept an input defining Zha's above-discussed "schedule" including a "frequency" and "times during the day" for taking snapshots and backing up transactional/transaction logs. Zha, 11:33-54; *supra* §VI.B. Fields related to database protection in database-provisioning UIs were well-known. Bhattacharjee, ¶¶144-145; RDS, 16-17, 187 ("backup window" for taking "snapshots"); Oracle, 7 ("Backup Destination"); Google, 2-3. Additionally, POSAs understood that accepting a protection schedule via Sivasubramanian's database-provisioning UIs would beneficially allow the database to be protected as soon as it is provisioned, since the schedule would already be specified. Bhattacharjee, ¶146. POSAs would have reasonably expected success with such UI implementations, because including fields related to database protection in database-provisioning UIs was well-known, as discussed above, and because customizing Sivasubramanian's UIs was within a POSA's skill (*supra* §VI.A.2). Bhattacharjee, ¶147

POSAs would have found it obvious to add parameters for specifying the protection schedule to Sivasubramanian’s command-line database-provisioning commands (*supra* §VI.A.1.a.ii) for the same reasons. Bhattacharjee, ¶148.

D. Claim-by-Claim Analysis

1. Claim 1

a. [1PRE]: “A method comprising”⁴

Sivasubramanian discloses a “computer-implemented *method* of controlling... a configuration aspect of a data environment using a control environment.” Sivasubramanian, claims 1, 4, 15; *see also* 2:17-18. As explained below, Sivasubramanian’s *method*, in view of Zha’s teachings as described *supra* §VI.C, *comprises* the remaining steps recited in claim 1. Bhattacharjee, ¶149.

b. [1A1]: “receiving, by a processor executing computer-readable instructions stored on a memory...”

Sivasubramanian’s invention can be implemented with a “computing device” including “*computer readable media*” “*storing*” “*computer readable instructions*,” examples of such media include various “*memory*” types.

Sivasubramanian, 22:33-37, 23:42-48, 23:61-24:8; Bhattacharjee, ¶150.

Sivasubramanian also says the “control plane” includes “computer-executable

⁴ Claim elements are referenced by the reference numbers defined *infra* §X (Claim Listing).

software” (*i.e. instructions*) and Sivasubramanian includes claims reciting a “control environment” operating “under control of” a “computer system[] configured with “executable *instructions*.” Sivasubramanian, 5:35-39, 24:20-24, 25:4-8, 26:23-26; Bhattacharjee, ¶150. Thus, Sivasubramanian discloses implementing its “control plane” via *computer-readable instructions stored on a memory*. Bhattacharjee, ¶150.

Furthermore, Sivasubramanian teaches *a processor executing the* aforementioned *instructions* implementing the “control plane.” As discussed above, the *instructions* are Sivasubramanian’s above-discussed “computer-readable instructions” for the control plane; these instructions are intended to be executed by a processor. Bhattacharjee, ¶151; Sivasubramanian, 27:18-35 (claim 20 reciting “system for controlling a data environment using a control environment” comprising “processor” and “instructions that, when executed by” the “processor” “cause the system” to perform actions), 5:35-39 (control plane includes “computer-executable software”).

As explained below, all steps recited in [1A2]-[1E] are performed as part of provisioning a database using the control plane’s database-provisioning UIs; thus, since the *processor* executes the control-plane instructions as discussed above, *the processor* performs the claim’s recited tasks, including *receiving* the required *user request* ([1A2]). Bhattacharjee, ¶152.

c. [1A2]: “a user request for provisioning a source database...”

i. GUI

As explained above generally in §VI.A, and more specifically in §VI.A.1 and §VI.A.1.a, Sivasubramanian discloses starting the process for *provisioning* a *database* in response to a *user request*. Sivasubramanian’s Figure 5 “illustrates an example of a display... to make calls into the control plane” when the user “**wants to provision a new data instance**” (*i.e., a database, supra* §VI.A.1).

Sivasubramanian, 12:38-40, 12:52-55. As explained *supra* §VI.A.1.a.i, control-plane users could either: (1) select the “Provision” option to *provision* an existing *database*; or (2) select the “Create” option to *provision* a newly-created database, as discussed *supra* §VI.A.1. Bhattacharjee, ¶¶153-154.

Web Browser

FILE EDIT VIEW FAVORITES TOOLS HELP

Address GO

Control interface page

Repository name

Task ☒ Provision ☐ Create ☐ Delete ☐ Modify

Database engine Version:

Storage capacity Units

Partition scheme

Security scheme Version:

Access key Version:

Sivasubramanian, Fig. 5 (annotated)

The '336 patent describes a “*source database*” as a database from which snapshots are taken to create backups. *See, e.g.*, EX1001, 4:46-57, 5:20-23; Bhattacharjee, ¶155. As discussed below for Element [1E], both the “Provision” and “Create” tasks in the GUI set up the database being provisioned to have snapshots captured from it to create backups, according to either Sivasubramanian+Zha implementation discussed *supra* §VI.C; thus, in Sivasubramanian+Zha, the *database* that is *provisioned* is a *source database*.

Bhattacharjee, ¶155; *see also* Zha, 1:66-2:6 (referring to backed-up database as “*source database*”).

Sivasubramanian describes Figure 5’s GUI as an “interface page” that a “*user*... will access... through the Internet.” Sivasubramanian, 12:48-52. POSAs understood that the *user* “access[es]” the page through the Internet by making a “*request[]*” that the control plane “*receives*.” Sivasubramanian, 5:35-41 (“[t]he control plane... includes a Web services layer” that “*receiv[es]* Web services calls or *requests*”); Bhattacharjee, ¶156. This “user” request for “access” to the Figure 5 database-provisioning UI is a *user request for provisioning a database*, since this request invokes Figure 5’s database-provisioning UI. Bhattacharjee, ¶156; EX1001, 13:2-5 (“[i]n response to the *user request* for a database *provisioning* service, the database engine... may activate the database *provisioning* system[.]”). Moreover, Sivasubramanian says Figure 5 is example “interface” that lets a “*user*... submit *requests* to the control plane,” which the control plane “*receiv[es]*”; thus, the request submitted using the Create or Provision task is *also* a *user request* as claimed. Sivasubramanian, 12:49-55, 10:15-17, claims 1, 4, 20, 25; Bhattacharjee, ¶157.

ii. Command Line

Sivasubramanian’s “*rds-create-database*” command *provisions* a new database, while the obvious “*rds-provision-database*” command *provisions* an

existing database. Sivasubramanian, 12:53-13:5; *supra* §VI.A.1.a.ii. As discussed above, the database is a *source database*. Sivasubramanian says the command-line is an alternative example to the GUI of an “interface” that lets a “*user*... submit *requests* to the control plane”; thus, the “rds-create-database” and “rds-provision-database” commands are each a *user request* as recited that the control plane *receives*. Sivasubramanian, 12:49-67; *supra* §VI.A.1.a.ii; Bhattacharjee, ¶158.

d. [1A3]: “in a hyperconverged infrastructure system”

“Hyperconverged” appears only in the ’336 patent’s title, and is nowhere in the specification. Bhattacharjee, ¶159.

Sivasubramanian’s databases are “create[d]” and “provision[ed]” *in* the “data plane.” Sivasubramanian, 7:3-9, 2:40-44, 5:15-20, 12:55-66, 13:28-29; Bhattacharjee, ¶160.

Sivasubramanian states that “*in one embodiment*,” the “data plane” is a “cloud” system. Sivasubramanian, 9:9-13. However, Sivasubramanian does not *require* that the data plane be a cloud. *E.g.*, Sivasubramanian, 24:19-28:56 (Sivasubramanian’s claims not requiring cloud); Bhattacharjee, ¶161. Rather, POSAs understood that Sivasubramanian is directed towards automating workflows but is flexible regarding the environment where workflows are performed. Sivasubramanian, 4:48-57 (“aspects of [Sivasubramanian’s] various embodiments can be used with any appropriate services available or offered over a

network in an electronic environment”), 3:10-25, 9:35-61, 15:45-50, 23:17-24; Bhattacharjee, ¶162.

One well-known alternative to a cloud-based system was a “hyperconverged infrastructure” (HCI) system. EX1036, [0003] (“Hyper Converged Infrastructure (HCI) combines computing applications and storage applications into a single infrastructure.”); EX1037, 1:13-33; Weissman, [0005] (cited by examiner, EX1002, 256); Bhattacharjee, ¶163. Some prior art described a “hyperconverged” infrastructure as an infrastructure type used by private entities where a single apparatus includes servers that are each configurable to perform different roles as needed, such as computation, storage, or networking. Srinivasan (EX1038), 1:15-27; Bhattacharjee, ¶164.

In Sivasubramanian+Zha, POSAs would have found it obvious to implement Sivasubramanian’s control plane to provision databases located *in* a “data plane” implemented as a *hyperconverged infrastructure system*. Bhattacharjee, ¶165. As noted above, HCI systems were known. Furthermore, it was known to deploy databases in hyperconverged systems. EX1039, [0242] (discussing “hyperconverged database type management system”); EX1040, [0089]; Srinivasan 6:46-53, Fig. 2; Bhattacharjee, ¶166. POSAs would have been motivated to implement the data plane as an HCI system as an alternative to a cloud to achieve the known benefits of HCI, such as resource sharing, flexibility,

and ease of management. Bhattacharjee, ¶167; EX1041, [0002]; EX1042, 1:5-10; EX1043, 1:6-20. POSAs would reasonably have expected success because HCI systems and their use for databases were well-known, as discussed above. Bhattacharjee, ¶168.

In the resulting implementation of Sivasubramanian+Zha, since the data plane is a hyperconverged infrastructure system, the *user request* (*supra* §VI.D.1.c ([1A2])) is *for provisioning a source database in a hyperconverged infrastructure system*. Bhattacharjee, ¶169.

e. [1B]: “receiving, by the processor, selection of a database engine type”

i. GUI

Sivasubramanian’s Fig. 5 (annotated below) shows *selection*, via a drop-down menu (element 506), of a “***Database engine***” type, in this case “MySQL,” for the “Provision” task. Bhattacharjee, ¶170; EX1001, 14:14-25; Sivasubramanian, 2:44-47; EX1101.

Web Browser

FILE EDIT VIEW FAVORITES TOOLS HELP

Address GO

Control interface page

Repository name

Task ☒ Provision ☐ Create ☐ Delete ☐ Modify

Database engine Version:

Storage capacity Units

Partition scheme

Security scheme Version:

Access key Version:

FIG. 5

POSAs understood that Figure 5’s GUI would similarly display a field for “database engine” *selection* when the “Create” task is selected, and would *receive [a] selection of a database engine type* via that field. *Supra* §VI.A.1.a.i; Bhattacharjee, ¶¶171-172; Sivasubramanian, 5:59-62 (referencing “[w]eb service frontend features such as... validating user input...”), 12:55-57 (Fig. 5 UI lets user “enter information”), 6:39-41, 10:37-40, 12:59-61, 19:25-31.

ii. Command Line

Sivasubramanian's "rds-create-database" command-line command, which provisions a newly-created database (*supra* §VI.A.1.a.ii), accepts the parameter "engine," which POSAs understood is a *database engine type*, since the exemplary value shown is "MySQL." Sivasubramanian, 12:64-13:5, Bhattacharjee, ¶173. POSAs would have found it obvious to implement an "rds-provision-database" command to also accept the "engine" parameter to match the "Provision" task in the GUI, and had the skill to do so. *Supra* §§VI.A.1.a.ii, VI.A.2; Bhattacharjee, ¶173. POSAs further understood the "engine" parameter is a *selection* because the user must select an engine type to enter the corresponding value for the "engine" parameter. Bhattacharjee, ¶173. POSAs also understood that the control plane *receives* this selection because it receives the command-line commands, as explained *supra* §VI.D.1.c.ii ([1A2]). Bhattacharjee, ¶173.

f. [1C]: "receiving, by the processor, selection of a Service Level Agreement ("SLA") and a protection schedule"

i. GUI

The '336 patent states that an "*SLA is an agreement...* that outlines, among other things, the protection scope of the database. The protection scope defines for how long data from the database being created or registered is retained." EX1001, 19:12-23.

Sivasubramanian+Zha’s graphical database-provisioning UI includes a UI element enabling *selection* of an *SLA* that specifies snapshot and transaction-log retention periods, and a UI element accepting a user-specified “schedule,” as Zha teaches. Zha, 11:33-44; *supra* §§VI.C.1-VI.C.2. As explained *infra* §VI.D.1.j ([1G]), this “schedule” *comprises a first and second frequency* as the claim recites. Thus, in Sivasubramanian+Zha, Sivasubramanian’s GUI *receives selection of a Service Level Agreement (SLA) and a protection schedule*. Bhattacharjee, ¶¶174-175.

ii. Command Line

In Sivasubramanian+Zha, Sivasubramanian’s command-line database-provisioning UI commands includes parameters for specifying a *selection* of an *SLA* and a “schedule,” as Zha teaches. *See supra* §§VI.C.1-VI.C.2. Thus, in Sivasubramanian+Zha, Sivasubramanian’s command-line database-provisioning UI commands *receive selection of a Service Level Agreement (SLA) and a protection schedule*. Bhattacharjee, ¶176.

g. [1D]: “provisioning, by the processor, the source database based upon the database engine type; and”

In Sivasubramanian+Zha’s GUI, the *database engine type* selection is received via the “Provision” and “Create” tasks in Figure 5’s GUI. *Supra* [1B] (§VI.D.1.e.i). Both tasks *provision the source database*. *Supra* [1A2] (§VI.D.1.c.i). Furthermore, in Sivasubramanian+Zha’s command-line interface,

the *database engine type* selection is received as a parameter of “rds-create-database” and “rds-provision-database” commands. *Supra* [1B] (§VI.D.1.e.ii). These commands also *provision the source database*. *Supra* [1A2] (§VI.D.1.c.ii). POSAs understood that the *source database* is *provisioned based upon* the selected *engine type*. See, e.g., Sivasubramanian, 6:39-50, 6:59-7:3, 12:58-62 (referring to the “Database engine” field labeled element 506 in Figure 5 as a “database engine *to be use[d]*” for the “action *to be performed*”), Fig. 5; Bhattacharjee, ¶¶177-178.

h. [1E]: “setting up, by the processor, automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database”

Sivasubramanian’s *processor* in Sivasubramanian+Zha receives a selection of an *SLA* and *protection schedule* via either the GUI or command-line interface for provisioning the *source database*. *Supra* [1C]. The *protection schedule* defines the frequencies at which the *snapshots* and *transactional logs*, respectively, are *captured*. *Infra* [1G]. And the *SLA* defines the durations for which the snapshots and logs are retained. *Supra* [1C], §VI.C.1.

In Sivasubramanian+Zha, snapshots and transactional logs are captured at user-defined frequencies and retained for user-defined retention periods, as Zha teaches. *Supra* §VI.C. Because the retention periods and frequencies are defined in the *SLA* and *protection schedule*, respectively, as discussed above, POSAs

understood that the *capturing of snapshots and transactional logs* is based upon the *SLA and protection schedule*. Bhattacharjee, ¶¶179-180.

Furthermore, the capturing is *automatic*. Zha teaches that the capturing of the snapshots and transaction logs each occur at a user-defined “frequency.” Zha, 11:33-44. POSAs would have understood that defining a “frequency” at which an event occurs causes that event to occur *automatically, i.e.,* without user involvement. Bhattacharjee, ¶181; EX1105, 109-110 (“automatic” means “given or imposed as a necessary and inevitable result of a fixed rule or particular set of circumstances”).

At minimum, POSAs would have found automatic capturing of snapshots and transactional logs obvious. Sivasubramanian discloses that the control plane can “*automatically* perform actions in the data plane.” Sivasubramanian, 2:3-5. Automatic capturing of information from a database was well-known. *E.g.*, RDS, 520, 647 (“automated DB snapshot”); Simek, Abstract (disclosing “automated creation... of database snapshots”). Furthermore, POSAs understood that automation beneficially reduced the need for user intervention, and would have been motivated to implement automatic snapshot capture in Sivasubramanian to achieve this benefit. Bhattacharjee, ¶¶182-183.

Additionally, POSAs understood that the purpose of the aforementioned *automatic capturing* is for *protecting the source database*. The ’336 patent says

“[p]rotecting a database means replicating a state of the database for creating a fully functional copy of the database”; this *“[r]eplicating... may involve creating* fully functional clones (e.g., *back-ups*),” which may in turn “be created by periodically capturing snapshots of the source database.” EX1001, 4:46-58; Bhattacharjee, ¶184. POSAs understood that in Sivasubramanian+Zha, the snapshot and log data is captured and retained for the purpose of “replicating a state of the database for creating a fully functional copy of the database” (EX1001, 4:46-58)—*i.e.*, for the purpose of *protecting the source database*. Bhattacharjee, ¶¶184-186; *see, e.g.*, Sivasubramanian, 15:1-15 (discussing “clon[ing]” database with “snapshots”), 19:20-20:1 (discussing “approach for backing up data” in database using snapshots and logs); Zha, 6:49-58, 17:63-66, & 25:24-34 (discussing process of creating virtual database with snapshots and logs to replicate source database); *supra* §§VI.B, VI.C.

Finally, Sivasubramanian teaches that the *setting up* of the automatic capturing is done *by the processor*, because the *processor* executes the control-plane instructions and the control plane controls the “recovery and backup” workflow. *Supra* [1A1]; Sivasubramanian, 6:31-42, 19:32-64; Bhattacharjee, ¶187.

i. [1F]: “wherein the transactional logs record transactions occurring on the source database since a most recent snapshot”

Sivasubramanian’s “logs,” which are *transactional logs* used to “replay” “transactions” (*supra* [1E], §VI.A.1.b; Sivasubramanian, 19:65-20:1), *record* transactions. Bhattacharjee, ¶188; EX1102, 1762-1763; Priebe, 1:19-54; EX1103, [0002]; EX1104, [0001]. POSAs understood, or at least would have found obvious, that the logs record transactions *since a most recent snapshot* because (1) Sivasubramanian refers broadly to the logs replaying “transactions” without specifying a stopping point and (2) Sivasubramanian discloses its “approach” for backing up a database backs up both “data files” and “logs,” indicating that logs record transactions occurring after snapshots (otherwise logs would be superfluous). Sivasubramanian, 19:20-64; Bhattacharjee, ¶188.

Additionally, in Sivasubramanian+Zha, Zha also teaches transactional logs *recording transactions since a most recent snapshot*. Zha, 10:6-11, 17:63-66 (describing using transaction/transactional logs “to reconstruct a copy of a database... corresponding to times in the past in between the times” when database snapshots were taken); Bhattacharjee, ¶189.

Finally, in both Sivasubramanian+Zha variants, the *transaction logs* record transactions *on the source database* because transaction logging is setup on the

source database to protect the source database when the source database is provisioned. *Supra* §VI.C, [1E]; Bhattacharjee, ¶188.

- j. **[1G]: “and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.”**

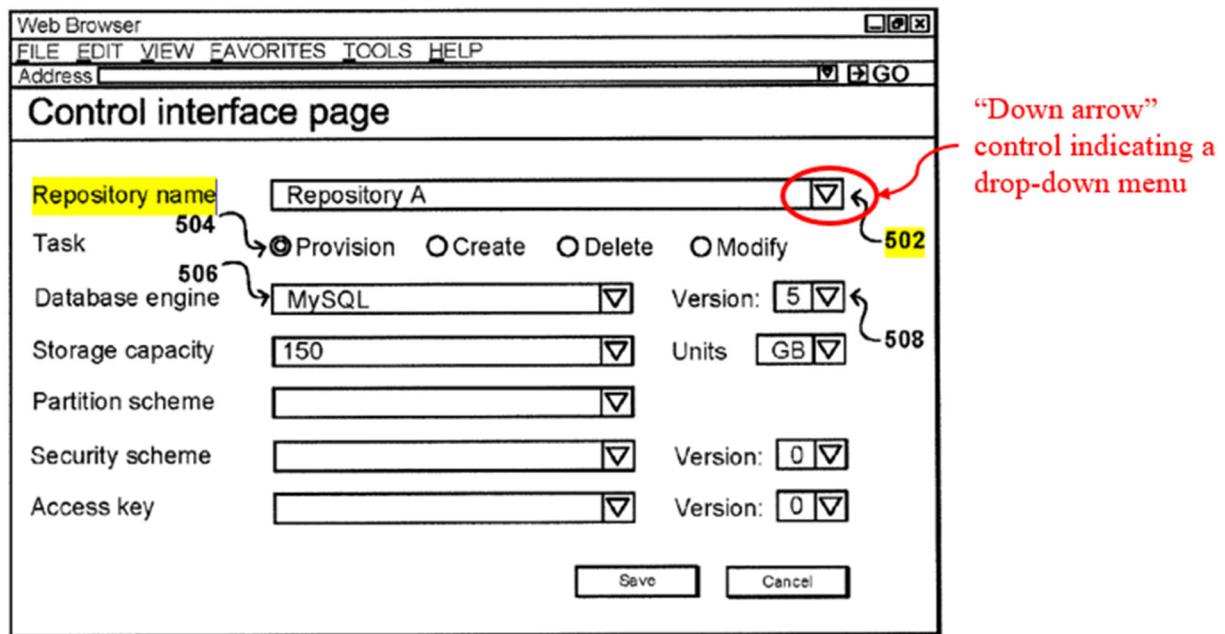
In Sivasubramanian+Zha, Sivasubramanian’s database-provisioning UIs accept a user-specified “*schedule*” as Zha teaches. *Supra* §VI.C.2, [1C]; Zha, 11:33-44. This “schedule” includes (1) a “*frequency*” for “retriev[ing] PIT copies,” *i.e.*, *capturing* snapshots, and (2) a “*frequency*” for “retriev[ing] updates to online transaction logs,” *i.e.*, *capturing* transaction/transactional logs. Zha, 11:33-44, 15:21-24; Bhattacharjee, ¶¶190-192; EX1001, 5:4-8; Heritage (EX1009), 277. Therefore, the user-specified “schedule” is a *protection schedule comprising a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured*. POSAs understood that the snapshots and transactional logs are captured *for protecting the source database* for the same reasons discussed *supra* [1E]. Bhattacharjee, ¶193

2. Claim 5

Sivasubramanian *provisions the source database* as part of the database-creation task launched from either Figure 5’s “Create” GUI or the “rds-create-database” command. *Supra* [1D]. In either scenario, *the provisioning comprises creating a new database as the source database*. Bhattacharjee, ¶194.

3. Claim 6

Sivasubramanian’s “Provision” task in Figure 5 lets the user *provision* a *source database* that is an *existing database* identified using the “Repository name” drop-down menu. Sivasubramanian, Fig. 5; *supra* §§VI.A.1.a.i, [1A2]. POSAs would likewise have found it obvious, and would reasonably expect success, to implement an “rds-provision-database” task (*supra* §VI.A.1.a.ii) to *provision* an *existing* database identified with the “name” command-line parameter. Sivasubramanian, 13:1-5; *supra* §VI.A.2; Bhattacharjee, ¶195-196.



Sivasubramanian, Detail of Fig. 5 (annotated)

In Sivasubramanian+Zha, Sivasubramanian’s database-provisioning UIs accept an SLA selection and protection schedule as Zha teaches. *Supra* §§VI.C, [1C]. POSAs would have understood that setting a protection schedule and SLA

for an existing database *registers the existing database as the source database*, because this action sets up the existing database as the source database for capturing snapshots. *Supra* [1C], [1E]; Bhattacharjee, ¶196; EX1001, 4:29-34, 25:43-67, FIGS. 5A-5E (describing “registering” existing database by providing connection information for the database and then specifying an SLA and protection schedule for it).

The ’336 patent says “existing databases” that are registered “*may* have been created outside of the database system,” indicating this is not *required*. If claim 6 is found to require it, however, POSAs would have found it obvious to implement Sivasubramanian’s database-provisioning UIs to allow a control-plane user to *register* a previously-unknown *existing database*. POSAs understood this would beneficially enable performing Sivasubramanian’s workflows, including the “recovery and backup” workflow, on existing databases, replacing “tedious manual procedures” for those databases. Sivasubramanian, 1:46-52, 6:31-42; Bhattacharjee, ¶197. POSAs would have reasonably expected success with such a UI implementation. Sivasubramanian discloses that (1) the control plane accepts input of parameters used for database access, such as DNS address and port, and (2) the control plane can “provision and manage” *pre-existing* databases in a data plane controlled by a “provider” separate from the control plane’s “provider.” Sivasubramanian, 6:59-67, 9:37-44, 11:40-52; Bhattacharjee, ¶197. Implementing

Sivasubramanian's UIs to accept these parameters was within a POSA's skill.

Supra §VI.A.2.

4. Claim 7

In Sivasubramanian+Zha, the user defines both (1) a “*frequency*” for taking snapshots and (2) a “*frequency*” for backing up transaction/transactional logs.

Zha, 11:33-44; *supra* §VI.C. These frequencies satisfy the claimed *first* and *second* frequency, respectively. *Supra* [1G]. POSAs understood that these frequencies *are the same* if the user chooses the same values for them.

Bhattacharjee, ¶198. POSAs would have found it obvious for a user to set the same frequency because this was one of only two options, *i.e.*, same or not, and because setting the same frequency beneficially reduced the need for a user to pick two different frequencies to select. Bhattacharjee, ¶198.

5. Claim 8

a. [8PRE]-[8A]

Sivasubramanian's Figure 5 (annotated below) shows the “*Database engine*” type (element 506) being selected from a drop-down menu. *Supra* [1B] (§VI.D.1.e.i). POSAs understood that such drop-down menus are used to *present a group* of options to select from. Bhattacharjee, ¶199; EX1101. Sivasubramanian's control plane “*support[s]* several different database engines,” *i.e.*, *database engine types*. Sivasubramanian, 9:61-64; *supra* [1B]. In Sivasubramanian+Zha, POSAs would have found it obvious that the database engine types the control plane

“support[s]” would be engine types supported by the *hyperconverged infrastructure system*, since the control plane provisions databases to run on the hyperconverged infrastructure system. *Supra* [1A3]VI.D.1.d; Bhattacharjee, ¶200. POSAs would have found it obvious to *present* this *group* for selection in the Figure 5 GUI because presenting the supported group beneficially allows the option of provisioning of databases running on any of the supported engine types. Bhattacharjee, ¶200. Because the *processor* executes the control plane instructions (*supra* §VI.D.1.b ([1A1])), the *presenting* is done *by the processor*.

b. [8B]

Figure 5 shows *receiving, by the processor, the selection of the “MySQL” database engine type* from the “Database engine” drop down menu. *Supra* [1B] (§VI.D.1.e.i). This selection is *from the group of database engine types* in the menu. *Supra* [8PRE]-[8A]; Bhattacharjee, ¶201.

6. Claim 10

In Sivasubramanian+Zha, SLAs are differentiated from each other by *levels* (*supra* §VI.C.1) and the *processor receives the selection* of an SLA via Sivasubramanian’s Figure 5 GUI. *Supra* [1C] (§VI.D.1.f.i). POSAs would have found it obvious to implement the control plane to *present* the various levels, *i.e., a plurality of levels*, to the user via a UI element for selection, *e.g., a drop-down menu* as disclosed in Sivasubramanian’s Figure 5 (*e.g. element 506*), and then

receive the selection of the *level*, since this would beneficially result in SLA selection for the database. Bhattacharjee, ¶202. Receiving such selection was within a POSA’s skill. *Supra* §VI.A.2. Because the *processor* executes the control plane instructions (*supra* [1A1]), the *presenting* and *receiving* are done *by the processor*

7. Claim 11

The ’336 patent uses “*time[-]/period*” in the context of snapshots to refer to a time of day at which a snapshot is captured. EX1001, 5:64-67, 20:17-24; Bhattacharjee, ¶203. The patent describes the “daily” snapshot as the snapshot taken at a “designated” daily time; this snapshot is stored for a longer duration than other snapshots taken that day (*e.g.*, 90 days v. 30). EX1001, 19:61-67, 21:4-12 (“GOLD” SLA). “Weekly,” “monthly,” and “quarterly” snapshots can be similarly designated. EX1001, 20:1-48; Bhattacharjee, ¶204.

In Sivasubramanian+Zha, Sivasubramanian’s UI accepts a “schedule” (*the protection schedule, see* [1C]) specifying “times during the day” for the “point in time” copies, *i.e.*, *snapshots*, as Zha teaches. Zha, 11:33-44; *supra* §VI.C. Thus, the *protection schedule* comprises a time period for designating at least one of a *daily snapshot*. Bhattacharjee, ¶205. This satisfies claim 11 if due to the claim language “at least one of... and,” the claim can be met by *designating* any listed item. *See also* EX1001, 39:31-40.

If the claim requires that *one* snapshot collected at *one* designated daily time be treated as “*the*” daily snapshot, this would have been obvious. Designating specific snapshots as daily (or weekly/monthly/quarterly) snapshots was well-known. Green (EX1047), [0163]-[0165]; Madany, [0023]; EX1048, [0060]; EX1049, [0164]; EX1050, [0053]; EX1051, [0025]; Bhattacharjee, ¶206. POSAs would have found it obvious to implement Sivasubramanian+Zha’s schedule UI (*supra* §VI.C.2) to additionally accept input of the daily snapshot time, as this was a known UI field that beneficially let users choose the best daily snapshot time based on their own circumstances. RDS, 16-17, 187 (daily “backup window” during which “snapshot” is taken); Gaurav, 11:14-20, Fig. 2; Bhattacharjee, ¶207. POSAs would have expected success because the UI implementation was within a POSA’s skill. *Supra* §VI.A.2.

If the claim requires *all* the recited items, this would have been obvious; it was known to specify time different periods for capturing snapshots for different snapshot frequencies, and this would have provided the same benefits discussed in the previous paragraph for daily snapshot time designation and would have been within a POSA’s skill. Bhattacharjee, ¶208; *see, e.g.*, Green, [0165]; Guarav, 10:20-22, 11:14-20, Fig. 2; *supra* §VI.A.2.

8. Claim 12

a. [12PRE]

Sivasubramanian describes “*/s/*systems and methods” that “provide a relational *database* service (‘RDS’)” that lets users “*configure relational databases,*” a task Sivasubramanian refers to as “*database management.*”

Sivasubramanian, 4:48-53. This *database management system*, as described in Sivasubramanian+Zha, includes the remaining components required in claim 12. Bhattacharjee, ¶¶209-210.

b. Limitations [12A1]-[12G]

The remaining limitations of claim 12 are obvious over Sivasubramanian+Zha for at least the reasons discussed for the prior elements indicated below:

Element	Prior Element
[12A1]	[1A1]
[12A2]	[1A1]
[12A3]	[1A2]-[1A3]
[12B]	[1B]
[12C]	[1C]
[12D]	[1D]
[12E]	[1E]
[12F]	[1F]
[12G]	[1G]

9. Claim 13

Sivasubramanian+Zha receives the user request, the selection of the database engine type, and the SLA via the Figure 5 GUI. *Supra* [1A]-[1C]:

The screenshot shows a web browser window titled "Web Browser" with a menu bar (FILE, EDIT, VIEW, FAVORITES, TOOLS, HELP) and an address bar. The main content area is titled "Control interface page". It contains the following fields and controls:

- Repository name: A text box containing "Repository A" with a dropdown arrow (502).
- Task: A group of radio buttons with labels "Provision" (selected), "Create", "Delete", and "Modify" (504).
- Database engine: A text box containing "MySQL" with a dropdown arrow (506).
- Version: A text box containing "5" with a dropdown arrow (508).
- Storage capacity: A text box containing "150" with a dropdown arrow.
- Units: A text box containing "GB" with a dropdown arrow.
- Partition scheme: A text box with a dropdown arrow.
- Security scheme: A text box with a dropdown arrow.
- Version: A text box containing "0" with a dropdown arrow.
- Access key: A text box with a dropdown arrow.
- Version: A text box containing "0" with a dropdown arrow.

At the bottom right, there are "Save" and "Cancel" buttons.

Sivasubramanian, Detail of Fig. 5

POSAs understood that a GUI like Figure 5 that lets a user provision and manage databases was commonly called a “*dashboard*.” Bhattacharjee, ¶¶211-212; *see, e.g.*, Nickolov (EX1062), [0123]-[0144], [0502]-[0507]; Karinta, [0297]-[0307], Fig. 8A.

10. Claim 14

Sivasubramanian+Zha teaches capturing snapshots and transactional logs of the source database according to the protection schedule. *Supra* §VI.C, [1C], [1E]. Sivasubramanian also describes using snapshots and transactional logs from a

“production instance” (*i.e.*, a *source database*) to create a “test instance” of the database that “is fully populated” with the data from the production instance. Sivasubramanian, 15:1-3. Sivasubramanian refers to this process as creating a “**clone**” of the production instance of the database (*i.e.*, a *cloned database from the source database based on the snapshots and the transactional logs*).

Sivasubramanian, 15:7-9; Bhattacharjee, ¶213.

Additionally, Zha describes using “**database blocks** retrieved by a point in time copy [*i.e.*, snapshot] manager” “combined with the **transaction logs** retrieved by the transaction log manager... to reconstruct” (*i.e.*, *create*) “**a copy of a database**” (*i.e.*, a *cloned database from the source database based on the snapshot and the transactional logs*). Zha, 10:6-12; Bhattacharjee, ¶214.

11. Claim 15

In Sivasubramanian+Zha, the user defines both (1) a “**frequency**” for taking snapshots and (2) a “**frequency**” for backing up transaction/transactional logs. Zha, 11:33-44; *supra* §VI.C. These frequencies satisfy the claimed *first* and *second* frequency, respectively. *Supra* [1G]. Since the user sets these frequencies, POSAs understood that these frequencies *are different* if the user chooses different frequencies. POSAs would have found it obvious for a user to set different frequencies because this was one of only two options, *i.e.*, same or not. Bhattacharjee, ¶215. Furthermore, it was known for transactional log backup

frequencies to be higher than snapshot frequencies (e.g., Tekade, [0041], [0045]-[0046], Fig. 6), and POSAs understood this arrangement beneficially allowed the database's state to be restored to points in time "between the times at which point-in-time copies [*i.e.*, snapshots, *see supra* §VI.B] are made," as Zha discloses. Zha, 10:6-12; Bhattacharjee, ¶215.

12. Claim 19

a. [19PRE]

Sivasubramanian states that its invention can be implemented with a "computing device" including "computer readable media" storing "information such as computer readable instructions," where the "media" includes "ROM" and "disk." Sivasubramanian, 22:33-37, 23:39-48, 23:61-24:8; *see also* 24:20-24, 25:4-8, 26:23-26; Bhattacharjee, ¶216. Thus, Sivasubramanian discloses implementing its "control plane" (described above) via a *non-transitory computer-readable media with computer-executable instructions embodied thereon*. Bhattacharjee, ¶¶216-220.

In Sivasubramanian, the aforementioned instructions are executed by a *processor in a hyperconverged infrastructure system* for the reasons explained *supra* [1A1], [1A3].

a. Limitations [19A]-[19G]

The remaining limitations of claim 19 are obvious over Sivasubramanian+Zha for at least the reasons discussed for the prior elements indicated in the table below:

Element	Prior Element
[19A]	[1A1]-[1A3]
[19B]	[1B]
[19C]	[1C]
[19D]	[1D]
[19E]	[1E]
[19F]	[1F]
[19G]	[1G]

13. Claim 20

Supra claims 5 and 6. Bhattacharjee, ¶219.

14. Claim 21

Supra claim 8. Bhattacharjee, ¶220.

VII. GROUND 2: CLAIMS 1-23 ARE UNPATENTABLE OVER SIVASUBRAMANIAN+SHEKAR+ZHA

A. Shekar (EX1007)

Shekar discloses a “[h]yperconverged infrastructure” (“HCI”) providing “storage” and “compute” capabilities. Shekar, Title, [0014], [0032], Fig. 1. Shekar’s goal is to provide a “storage device” with “high performance.” Shekar, [0005]. Bhattacharjee, ¶¶221-222.

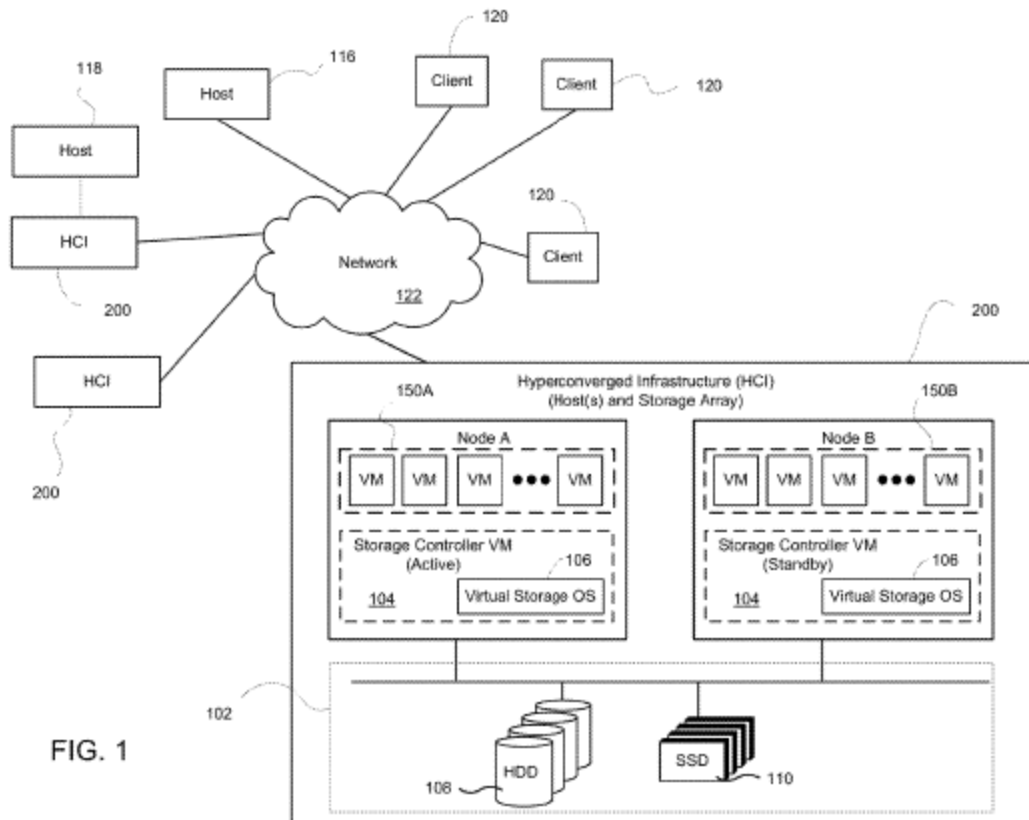


FIG. 1

Shekar, Fig. 1

Shekar's HCI includes two "nodes" that include a "virtual storage connect module" connecting the node to a "storage array"; the array "provide[s] storage services to one or more host applications executing on the corresponding HCI 200." Shekar, [0035], [0037], [0043], Fig. 2A; Bhattacharjee, ¶¶223-224. The applications run on "guest virtual machines [VMs]" executing on each node's hardware. Shekar, [0008], [0038], Fig. 2A; Bhattacharjee, ¶225. Each node also includes a "storage VM" (or "storage controller VM"). Shekar, [0033], Fig. 2A. The storage VMs on the two nodes "cooperatively manage access to the storage array." Shekar, [0033], [0040]-[0042]; Bhattacharjee, ¶226.

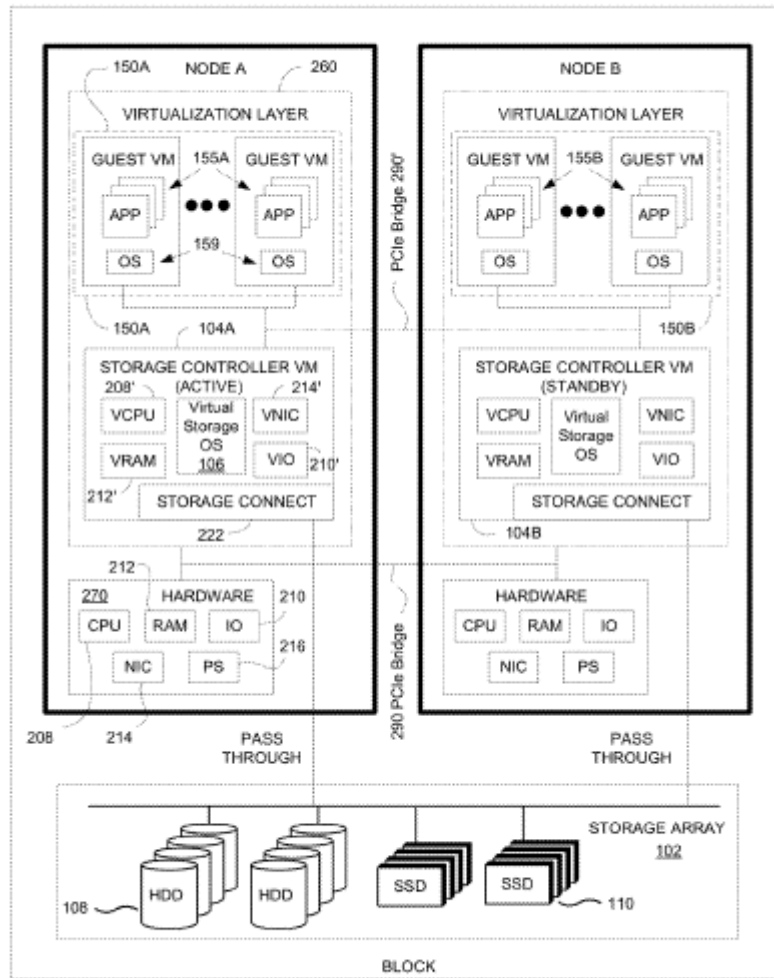


FIG. 2A

Shekar, Fig. 2A

B. Sivasubramanian+Shekar+Zha Combination

Sivasubramanian's control plane is flexible regarding data-plane type.

Supra Ground 1, [1A3]. In view of Shekar, POSAs would have been motivated to provide the option of a "data plane" implemented using one or more of Shekar's "HCIs," for multiple reasons. Bhattacharjee, ¶227. First, as explained, POSAs understood that hyperconverged infrastructure was a known option for

implementing a “data plane” with known benefits. *Supra* Ground 1, [1A3].

Second, POSAs would have been specifically motivated to use Shekar’s “HCI[s]” to beneficially achieve the “high performance” of Shekar’s HCIs in a “data plane.” Shekar, [0005]; Bhattacharjee, ¶¶228-229. Third, Shekar discloses that each HCI includes a “storage array” that provides “storage services” to “host applications” running in the HCI. Shekar, [0035]; *supra* §VII.A. POSA understood that “database[s]” such as those provisioned by Sivasubramanian (Sivasubramanian, 6:59-56; *supra* §VI.A.1) are applications that use storage services, and thus that would benefit from incorporation of Shekar’s HCI. Bhattacharjee, ¶230; Sivasubramanian, 6:43-58.

The resulting Sivasubramanian+Shekar system includes a “control plane” as taught by Sivasubramanian (*supra* §VI.A.1) that executes workflows in a data plane implemented using one or more of Shekar’s HCIs (*supra* §VII.A). In this system, databases, which are “applications” as noted above, run on “virtual machines” (“VMs”) running on HCIs, as taught in Shekar. Shekar, [0008]-[0010], [0032], [0035], [0038], [0040]; *supra* §VII.A. Additionally, as detailed *infra* claims 2 and 3, in the resulting system, Sivasubramanian’s graphical and command-line database-provisioning UIs are implemented to accept additional parameters for configuring the VMs. Bhattacharjee, ¶231.

Furthermore, POSAs would have found it obvious to incorporate Zha's teachings regarding user-defined snapshot and transaction-log collection frequencies, times during the day, and/or retention periods, and also incorporate UI elements to gather such information, into the Sivasubramanian+Shekar system for the same reasons as the Sivasubramanian+Zha combination. *Supra* §VI.C; Bhattacharjee, ¶232.

The resulting Sivasubramanian+Shekar+Zha combination is identical to the Sivasubramanian+Zha combination of Ground 1 plus two aspects: (1) the data plane is implemented using Shekar's HCIs, with databases running on VMs on the HCIs; and (2) additional UI elements are added for configuring VMs, as discussed above and detailed in Claims 2-3 below. Bhattacharjee, ¶233.

POSAs would reasonably have expected success in the above-described implementation for multiple reasons, as explained below.

Using hyperconverged systems for databases was known in the art. *Supra* §VI.D.1.d (Ground 1, [1A3]). Running databases on virtual machines ("VMs") was also well-known in the art. Uriel, [0047]; Winterfeldt (EX1067), [0021]; Jin-844, [0007]; Jin-297 (EX1069), 3:16-19; Bhattacharjee, ¶¶234-235. Additionally, POSAs understood that a VM emulates a physical computer system; thus, POSAs would have reasonably expected that running databases on VMs on Shekar's HCIs would not change the functions of the databases in any aspect relevant to

Sivasubramanian's control plane. Bhattacharjee, ¶236; Bhandari (EX1070), 1:10-17.

Moreover, POSAs understood that Shekar's "HCI" includes components that are functionally equivalent to components that Sivasubramanian describes as being used to run databases in the data plane. Bhattacharjee, ¶237. Shekar's HCI includes a "storage array" providing "storage services" to "host applications" running in the HCI. Shekar, [0035]; *supra* §VII.A. Similarly, Sivasubramanian discloses that a "data instance" for a database (*see supra* §VI.A.1), is "allocate[ed]" and "attach[ed]... to" "off-instance persistent storage." Sivasubramanian, 6:59-65. As another example, each of Shekar's "virtual host systems" (also called "VMs") on which "applications" run includes a "VCPU" (virtual CPU) and "virtual memory (RAM)." Shekar, [0046]. Similarly, Sivasubramanian discloses that "instances" for databases may have an "instance type" that "describes the CPU and memory capacity." Sivasubramanian, 11:49-52; Bhattacharjee, ¶237. Furthermore, Shekar explains that the nodes in an HCI include "one or more network interface cards (NICs)... for exchanging data packages through a network." Shekar, [0039]. Thus, POSAs understood that Sivasubramanian's control plane would be able to communicate with the databases running on VMs in the HCIs to execute workflows. Bhattacharjee, ¶¶238-239.

Finally, implementing Sivasubramanian’s UI to accept additional parameters was within a POSA’s skill. *Supra* §VI.A.2.

C. Claim-by-Claim Analysis

1. Claims 1, 5-8, 10-15, 19-21

As explained *supra* Ground 1, [1A3], Sivasubramanian discloses that databases are provisioned in the “data plane.” Sivasubramanian, 7:3-9, 2:40-44, 5:15-20, 12:55-66, 13:28-29. As explained *supra* §VII.B, Sivasubramanian+Shekar+Zha is identical to Sivasubramanian+Zha except that (1) the “data plane” may be implemented using Shekar’s HCIs, with the databases running on VMs running on the HCIs; and (2) the Sivasubramanian+Shekar+Zha UIs accept additional parameters. Bhattacharjee, ¶240. Shekar describes its HCI as an “integrated *system*.” Shekar, [0037]. Thus, in Sivasubramanian+Shekar+Zha, the source database is provisioned on a VM *in a hyperconverged infrastructure system*. Bhattacharjee, ¶241. Sivasubramanian+Shekar+Zha meets all remaining limitations of claim 1, and also meets claims 5-8, 10-15, and 19-21, for the same reasons given *supra* §VI for Sivasubramanian+Zha in Ground 1.

2. Claim 2

a. [2PRE]-[2A]

Shekar’s VMs have properties including “operating system” and amount of “memory (RAM).” Shekar [0046], Fig. 2B. Shekar’s VMs can also be

“configured.” Shekar, [0038]. Furthermore, it was known to customize the properties of VMs, including VMs on which databases ran. Bhandari, 1:11-15 (VMs “can be customized” with “computing resources” including “RAM,” and “operating system”); Winterfeldt, [0029]-[0035] (describing “[VM] templates” for VMs running databases); VanDerWalt (EX1071), [0158]-[0159]; Hardy (EX1072), [0066]; Jin-297, 1:19-25, 7:49-8:2; Bhattacharjee, ¶242.

Thus, in Sivasubramanian+Shekar+Zha, where a database runs on a VM in Shekar’s HCI (*supra* §VII.B), POSAs would have found it obvious to implement Sivasubramanian’s graphical and command-line database-provisioning UIs (*supra* §VI.A.1.a) to *receive a selection of a set of values* for VM properties, including OS and RAM, to customize the VM. POSAs would have been motivated to implement this functionality to beneficially give control-plane users flexibility to customize VMs based on the database’s needs, and such an implementation would have been within a POSA’s skill. *Supra* §VI.A.2; Bhattacharjee, ¶243.

The aforementioned values for VM properties like OS and memory are *parameters*. *E.g.*, EX1001, 16:51-54, 17:13-48; Bhandari, 8:19-24; Bhattacharjee, ¶244. Furthermore, that set of parameters is a *first set* distinct from other parameters like database engine type, SLA, or protection schedule. *Supra* Ground 1, [1B]-[1C]; Bhattacharjee, ¶244.

Because the *processor* executes the instructions for the control plane, the *receiving* is done *by the processor*. *Supra* Ground 1, [1A1]; Bhattacharjee, ¶245.

b. [2B]

As explained *supra* [2A], the *first set of parameters* are parameters for a virtual machine. Thus, POSAs would have found it obvious to implement Sivasubramanian+Shekar+Zha to *create a virtual machine based on the first set of parameters*. Bhattacharjee, ¶246.

As explained *supra* [2A], the parameters are received via the database-provisioning UIs. These UIs are used to provision the *source database*. *Supra* Ground 1, [1D]. In Sivasubramanian+Shekar+Zha, a database runs as an “application” on a VM running on an HCI, as taught in Shekar. Shekar, [0008]; *supra* §VII.B. In Shekar, “applications” such as databases “resid[e] on” the VMs on Shekar’s HCIs. Shekar, [0038], [0035], [0046], Fig. 2B (showing “applications 155” inside “VM”); Bhattacharjee, ¶¶247-248. Thus, in Sivasubramanian+Shekar+Zha, the *source database resides on* the created *virtual machine*. Bhattacharjee, ¶249.

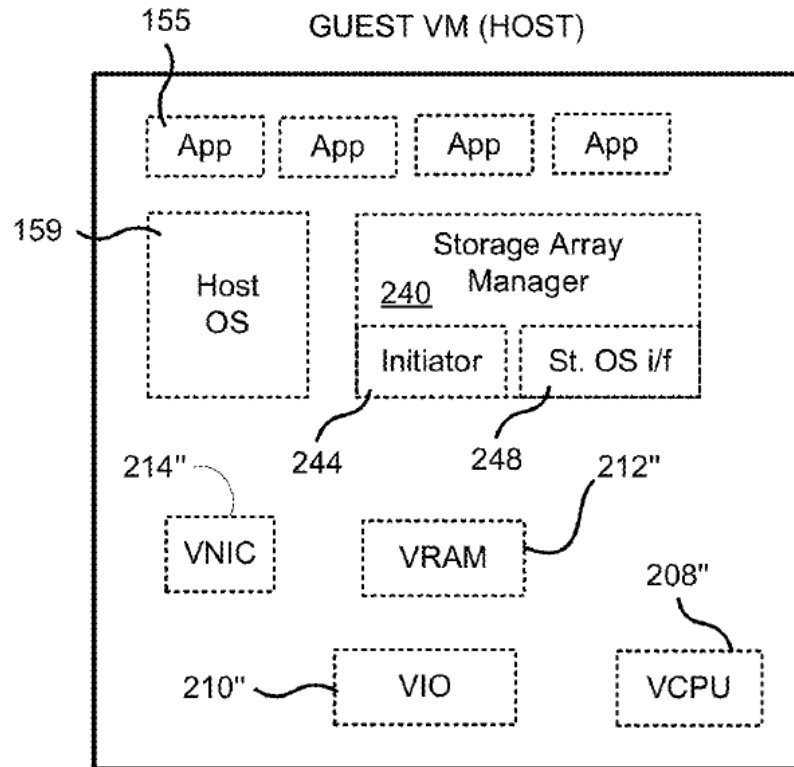


FIG. 2B

Shekar, Fig. 2B

The '336 patent describes a “database virtual machine” as a VM that “host[s]” a database, and describes a “*source database virtual machine*” as a virtual machine “associated with” the “storage” used by a source database. *See, e.g.,* EX1001, 3:32-49, 14:59-61; Bhattacharjee, ¶250.

Similarly, Shekar refers to VMs as “*host/s*” for applications, and each VM includes a “storage array manager” that allows “applications” to access the “storage array.” *E.g.*, Shekar, [0046]-[0047], Fig. 2B. In Sivasubramanian+Shekar+Zha, the databases running on Shekar’s VMs use Shekar’s storage array for storage. *Supra* §VII.B. Thus, the aforementioned virtual machine on which the source database resides is a *source database virtual machine*. Bhattacharjee, ¶251.

Because the *processor* executes the instructions for the control plane, the *creating* is done *by the processor*. *Supra* Ground 1, [1A1]; Bhattacharjee, ¶252.

3. Claim 3

a. “software profile”

The ’336 patent says a “*software profile* defines the software and operating system [OS] parameters for the database engine type that is selected” during the database “provisioning” process. EX1001, 17:13-19, 15:52-57; Bhattacharjee, ¶253. As explained *supra* [2A], the *first set of parameters* comprises the OS for the VM that the database of the selected engine type runs on. Additionally, Sivasubramanian discloses specifying “version information” for the *selected database engine type* in both the Figure 5 GUI and the command-line. Sivasubramanian, 12:57-13:5, Fig. 5; *supra* Ground 1, [1B]. Thus, the *first set of*

parameters further comprises the database version, and the OS and database version together comprise a *software profile*. Bhattacharjee, ¶253.

The screenshot shows a web browser window titled 'Web Browser' with a menu bar (FILE, EDIT, VIEW, FAVORITES, TOOLS, HELP) and an address bar. The main content area is titled 'Control interface page'. It contains the following form elements:

- Repository name:** A text input field containing 'Repository A'.
- Task:** A group of radio buttons with labels 'Provision', 'Create', 'Delete', and 'Modify'. The 'Provision' button is selected.
- Database engine:** A dropdown menu showing 'MySQL'.
- Storage capacity:** A text input field containing '150'.
- Units:** A dropdown menu showing 'GB'.
- Partition scheme:** A text input field.
- Security scheme:** A text input field.
- Access key:** A text input field.
- Version:** There are three version dropdowns: one for the database engine (showing '5'), one for the security scheme (showing '0'), and one for the access key (showing '0').
- Buttons:** 'Save' and 'Cancel' buttons at the bottom right.

Annotations with arrows point to the following elements:

- 502:** Points to the 'Provision' radio button.
- 504:** Points to the 'MySQL' dropdown menu.
- 506:** Points to the 'Storage capacity' text input field.
- 508:** Points to the 'Units' dropdown menu.

FIG. 5

Sivasubramanian, Fig. 5

If the claim requires the “software profile” be a predefined set of values, this would have been obvious. Defining sets of parameters values was well-known; such sets were often called “profiles” or “templates.” Bhattacharjee, ¶254; *see, e.g.*, Inskeep, Abstract, 1:31-34; RDS, 412; Winterfeldt, [0029]; Nickolov, [0734], Fig. 9; Hardy, [0066]; Nichele, [0002]; Jalagam, [0028]; Kundapur, [0022]; Babbar, [0050].

It was likewise well-known that allowing selection of a predefined “profile”/“template” beneficially increased user convenience by reducing the need

for users to choose and separately input parameter values. Bhattacharjee, ¶255; *see, e.g.*, Wren-Hilton, 1-2; Mikan, [0003]; Poulsen, 1:16-29; Miller, [0225]-[0226] & Fig. 16A; Kerpez, [0105].

POSAs understood that certain versions of applications like database engines were compatible with, or optimized for, specific OSs or versions thereof. *See, e.g.*, Zamir, [0027]; Jayaprakash, [0013]; Gallagher, [0012]; Sharma, [0031]; Heddaya, [0015]; MySQL, 1; Bhattacharjee, ¶256. Thus, in Sivasubramanian+Shekar+Zha, where the user may customize the source database's VM, POSAs would have found it obvious to implement Sivasubramanian's database creation and provisioning UIs so the *first set of parameters* received for configuring the VM (*supra* [2PRE]-[2A]) includes a predefined OS and supported database engine version, *i.e.*, a *software profile*; this would beneficially increase user convenience and was within a POSA's skill. Bhattacharjee, ¶256; *supra* §VI.A.2.

b. “compute profile”

A “compute profile defines the size/configuration of [a] source database VM,” *e.g.*, “the number of vCPUs [virtual CPUs]” or the “memory capacity.” EX1001, 17:42-46.

The *first set of parameters* comprises the amount of memory for the VM. *Supra* [2A]. Additionally, each VM in Shekar has a “VCPU.” Shekar, [0046]. In Sivasubramanian+Shekar+Zha, where the user may customize the source

database's VM, POSAs would have found it obvious to implement Sivasubramanian's database creation and provisioning UIs so the *first set of parameters* received for configuring the VM (*supra* [2PRE]-[2A]) includes a number of VCPUs and an amount of memory. Bhattacharjee, ¶¶257-258. Both were well-known configuration options for databases, and such an implementation was within a POSA's skill. *Supra* §VI.A.2; Bhattacharjee, ¶258; *see, e.g.*, RDS, 14, 104-105; Adogla, [0032]. If the claim requires a predefined set of these parameter values, this was a known UI feature that would have been obvious for the convenience reasons discussed above for software profiles. RDS, 14, 104-105; Adogla, [0032]; Bhattacharjee, ¶259.

c. “network profile”

Sivasubramanian's “example ‘create database’ workflow” includes “allocating and attaching a DNS address or other address, port, interface, or identifier,” and “a user is provided with the DNS address and a port address... to access the [database] instance.” Sivasubramanian, 6:59-67, 11:40-52; Bhattacharjee, ¶260. Additionally, Sivasubramanian's exemplary invocation of “rds-create-database” shows a “port” parameter being used. Sivasubramanian, 13:1-5. Thus, POSAs would have been motivated to implement Sivasubramanian's database-provisioning UIs in Sivasubramanian+Shekar+Zha so that the *first set of parameters* received for configuring the source database's VM

(*supra* [2PRE]-[2A]) includes the DNS address and port; this would beneficially enable access to the provisioned database by an entity “provide[d] with” that “DNS address and port.” Sivasubramanian, 6:59-67; Bhattacharjee, ¶261.

A “***network profile***” in the ’336 patent “identifies the network location of the database being created... to facilitate access to the database after creation.” EX1001, 17:31-33. POSAs understood that in Sivasubramanian, the “DNS address” identifies a database’s network “location.” Sivasubramanian, 2:40-44 (referencing “the ***DNS (domain name system) address or other location information***” for a database instance); Bhattacharjee, ¶¶262-263; Chung, [0003]. POSAs also understood that the “port” identifies the network location of a database, because an external entity “use[s]” a “port address” to “access” the database. Sivasubramanian, 6:65-67; Bhattacharjee, ¶263. Thus, the DNS address and port together constitute a *network profile*. Bhattacharjee, ¶263. If the claim requires a *network profile* be a predefined set of database-access-related parameter values, this was a known UI feature that would have been obvious for the convenience reasons discussed *supra* §VII.C.3.a. RDS, 412; Bhattacharjee, ¶264.

d. “database parameters profile”

“The database parameter profile defines the custom parameters that are applied to the database being created or registered.” EX1001, 17:51-53. The patent does not define “custom” parameters. Bhattacharjee, ¶265.

Sivasubramanian discloses that the database creation and provisioning UIs accept parameters including “Partition scheme,” “Security scheme,” “Access key,” and “password.” Sivasubramanian, 13:1-5, Fig. 5. Thus, POSAs would have been motivated to implement Sivasubramanian’s database creation and provisioning UIs in Sivasubramanian+Shekar+Zha so that the *first set of parameters* received for configuring the VM (*supra* [2PRE]-[2A]) include such parameters, which constitute a *database parameters profile*. Bhattacharjee, ¶266.

If claim 3 is interpreted to require that the database parameters profile be a single predefined set of values, this was a known UI feature that would have been obvious for the convenience reasons discussed above for software profiles.

Bhattacharjee, ¶267; RDS, 459-460.

4. Claim 4

The ’336 patent refers to the user specifying a “database size to determine how much storage is needed for storing” the database.” EX1001, 18:46-52, 4:18-22; Bhattacharjee, ¶268. The exemplary UI also includes a database “size” field. EX1001, 24:51-63, Fig. 4E. The patent describes an exemplary “cluster” in a “virtual computing system” that includes a “storage pool” containing “storage components” “configure[able] as virtual disks for access by...database VMs.” EX1001, 6:5-7, 8:18-22, Fig. 1. The “storage pool” may include “network-attached” or “direct-attached” storage. EX1001, 6:36-41; Bhattacharjee, ¶268.

POSAs understood that a “virtual disk” is a unit of storage allocated to a virtual machine that provides a virtual representation of physical storage. Cao (EX1089), 1:15-16; Wang (EX1090), 1:13-14; Fruchtman (EX1091), [0002]. Bhattacharjee, ¶269.

Sivasubramanian discloses the user of the control plane specifying a “*storage* capacity” or “*storage* size” for a database being created or provisioned. Sivasubramanian, 11:49-52, 13:1-5, Figure 5.

The screenshot shows a web browser window titled "Web Browser" with a menu bar (FILE, EDIT, VIEW, FAVORITES, TOOLS, HELP) and an address bar. The main content area is titled "Control interface page". It contains the following form elements:

- Repository name:** A dropdown menu showing "Repository A".
- Task:** A group of radio buttons with labels "Provision", "Create", "Delete", and "Modify". The "Provision" button is selected.
- Database engine:** A dropdown menu showing "MySQL".
- Storage capacity:** A text input field containing "150".
- Units:** A dropdown menu showing "GB".
- Version:** A dropdown menu showing "5".
- Partition scheme:** A dropdown menu.
- Security scheme:** A dropdown menu.
- Version:** A dropdown menu showing "0".
- Access key:** A dropdown menu.
- Version:** A dropdown menu showing "0".

At the bottom right, there are two buttons: "Save" and "Cancel".

Annotations with arrows point to specific elements:

- 504 points to the "Provision" radio button.
- 506 points to the "MySQL" dropdown menu.
- 502 points to the "Repository A" dropdown menu.
- 508 points to the "5" dropdown menu.

FIG. 5

Thus, POSAs understood that in Sivasubramanian+Shekar+Zha, the *source database*, which is provisioned as a result of using Sivasubramanian’s database-provisioning UIs (*supra* Ground 1, [1D]), has a *size*. Bhattacharjee, ¶¶270-271.

In Sivasubramanian+Shekar+Zha, the *source database* is provisioned on a virtual machine running on Shekar’s HCI. *Supra* §VII.C.1. In Shekar’s HCI, the VMs use a “storage array” for data storage. Shekar, [0032]. The array may be “local” to the HCI or accessible over a “network.” Shekar, [0034]-[0035]. Storage array access is provided by “virtual storage controllers” on the HCI. Shekar, [0040]. Bhattacharjee, ¶272.

POSAs understood that it was well-known and customary to use virtual disks to provide storage for virtual machines. Cao, 1:15-16; Wang, 1:13-14; Fruchtman, [0002]; EX1092, 1:23-26; EX1093, 7:36-39; Jin-297, 3:16-26; Bhattacharjee, ¶273. It was also well-known for databases to use virtual disks to store data. EX1094, [0003]; 1095, [0052]; EX1064, [0019]; EX1096, [0063]; Bhattacharjee, ¶273. POSAs understood that using virtual disks on shared storage such as Shekar’s storage array provided benefits such as “streamlin[ing] storage management and configuration” and “enabl[ing] virtualization management techniques.” EX1097, [0002]; EX1098, [0001]; Bhattacharjee, ¶273. Thus, POSAs would have found it obvious to implement Sivasubramanian+Shekar+Zha so that the *source database* running on a VM on one of Shekar’s HCIs used *virtual*

disks to store data; such an implementation would have been within a POSA's skill, as using virtual disks as storage with both databases and VMs was well-known in the art, as discussed above. Bhattacharjee, ¶273.

POSAs also understood that it was customary for virtual disks to have a certain size, such as "10 GB." EX1099, [0002]; *see also* Wang, 1:15-17; EX1100, [0003]; Bhattacharjee, ¶274. Sivasubramanian discloses examples of users selecting database sizes for the source database of 80 or 150 GB. Sivasubramanian, 12:38-13:5, Fig. 5. POSAs understood based on simple mathematics that it was obvious to *convert* the *size* of the source database into a *number of virtual disks* to ensure that enough virtual disks are allocated to store the database; *e.g.*, a minimum of eight 10-GB virtual disks would be needed for an 80-GB database. Bhattacharjee, ¶274. Since the *processor* in the control plane performs the provisioning, the processor also performs the *converting*. *Supra* Ground 1, [1A1], [1D]; Bhattacharjee, ¶274.

Finally, POSAs would have understood that the *processor* would configure the source database so that it can use the virtual disks as storage, since the point of specifying storage capacity when provisioning a database is so that the database can use that storage. Bhattacharjee, ¶275; Sivasubramanian, 6:59-65, claim 10. Since this allows the database and virtual disks to collectively achieve the

functionality of accessing data, this constitutes *associating the source database* with the *number of virtual disks*. EX1001, 38:44-51; Bhattacharjee, ¶275.

5. Claim 9

As discussed above for claim 3, the '336 patent refers to different types of “profiles” as each defining multiple parameters, *i.e., sets of parameters*. EX1001, 17:13-19, 15:52-57, 17:42-46, 17:31-33, 17:51-53. The patent refers to “sample” profiles as predefined profiles the user can choose from, in contrast to custom profiles. EX1001, 17:19-30.

In Sivasubramanian+Shekar+Zha, the *processor receives parameters* in the form of various profiles and *creates the source database virtual machine based [up]on those parameters*. *Supra* claims 2-3; Bhattacharjee, ¶¶276-277. POSAs would have found it obvious to implement the control plane in Sivasubramanian+Shekar+Zha to provide at least one *sample* profile, *i.e., sample set of parameters*, to the user, and then *receive the selection* of that sample, because providing predefined profiles would increase user convenience and was within a POSA’s skill. Bhattacharjee, ¶277; *supra* §VI.A.2. For example, POSAs would have been motivated to provide predefined sets of operating systems and corresponding supported database software versions to eliminate the need for the user to memorize which software version runs on which operating system. *Supra* claim 3; Bhattacharjee, ¶277.

6. Claim 16

As explained above, in Sivasubramanian+Shekar+Zha, the processor *executes the computer-readable instructions to receive a set of parameters, i.e. at least one parameter, for creating a source database virtual machine and configures the database to run on that virtual machine, i.e., associates the source database with the source database virtual machine.* *Supra*, claim 2; Bhattacharjee, ¶278.

7. Claim 17

As discussed above, in Sivasubramanian+Shekar+Zha, the UI receives an input of a *size of the source database.* *Supra* claim 4. Because *the processor executes the computer-readable instructions for the control plane (supra* Ground 1, [1A1]), *the processor executes the instructions to receive the size.* Bhattacharjee, ¶279. Additionally, as discussed *supra* claim 4, the processor *converts the size into a number of virtual disks to be associated with the source database.*

8. Claim 18

As explained above, in Sivasubramanian+Shekar+Zha, the *source database is created on a source database virtual machine* that is a VM on Shekar's HCI. *Supra* §VII.B ([2B]); Bhattacharjee, ¶280. As an alternative to creating a new VM as discussed *supra* claim 2, POSAs would have found it obvious to implement Sivasubramanian+Shekar+Zha so that Sivasubramanian's UI allows the user to selecting an existing *source database virtual machine.* UIs that allowed a user to

select an existing VM for deployment of a database or another application were known in the art, and allowing the user to pick an existing VM would beneficially give the user the option of avoiding the added steps of creating a new VM; such a UI was within a POSA's skill. *E.g.*, VanDerWalt, [0135]-[0138], Figs. 18-19; *supra* §VI.A.2; Bhattacharjee, ¶281.

9. Claim 22

Supra claim 9. Bhattacharjee, ¶282.

10. Claim 23

Supra claim 10. Bhattacharjee, ¶283.

VIII. NO BASIS EXISTS FOR DISCRETIONARY DENIAL

A. 315(a)

This Petition presents “compelling evidence of unpatentability,” which should end the *Fintiv* analysis. Director's Interim Procedure for Discretionary Denials, 5 (June 21, 2022) (“Interim Procedure”) (discussing *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential)).

If considered, the *Fintiv* factors weigh against denial.

1. Stay Potential

This factor is neutral because no stay in favor of IPR has yet been requested, so there is no “specific evidence” how the court might rule. *Sand Revolution II v. Continental Intermodal Group-Trucking*, IPR2019-01393, Paper 24, 7 (June 16, 2020) (informative).

2. Trial Timing

This weighs against denial. *Fintiv*, 3 & n.3. No trial has yet been set. Given the Northern District of California’s median time to trial (47.9 months)⁵, institution would mean FWD before trial, further favoring institution. Interim Procedure, 8-9 & 4 n.12. Petitioner diligently filed this Petition roughly eight months after complaint service, before receiving infringement contentions, and before answering (EXs. 1056-1057). *Apple v. Seven Networks*, IPR2020-00156, Paper 10, 9 & n.8 (June 15, 2020) (“*Seven*”).

3. Litigation Investment

This weighs against denial. Petitioner filed this petition before any substantive orders in the litigation or significant investment. *Seven*, 11-12 (quoting *Fintiv*, 11).

4. Issue overlap

This weighs against denial. Petitioner challenges all claims; PO has not served contentions. The Board should institute to enhance patent system integrity and meet Petitioner’s interest in resolving invalidity challenges on all claims. *Seven*, 18.

⁵ See <https://www.uscourts.gov/file/79295/download>, at 9.

5. Litigation Defendants

This is neutral; Petitioner is a litigation defendant, which is not unusual. *Sand Revolution II*, 12-13.

6. Other Circumstances

The Petition’s “strong showing on the merits” weighs against denial. *Seven*, 20.

B. 325(d)

Considering the two-part framework discussed in *Advanced Bionics, LLC v. Med-El Elektromedizinische Gerate GMBH*, IPR2019-01469, Paper 6 (Feb. 13, 2020) (precedential), the Board should not exercise its §325(d) discretion to deny institution.

1. Part One

As the Director recently noted in a notice of proposed rulemaking, “application of section 325(d) should be limited to situations in which the prior art or arguments were *meaningfully* addressed by the Office”—meaning that the Office “*articulated* its consideration of the art or arguments *in the record*.” 89 Fed. Reg. 28693, 28700 (Apr. 19, 2024). The record here lacks any such articulation. *See* EX1002. Although Zha and a descendant of Sivasubramanian are listed on an IDS signed by the examiner (EX1002, 239-240), the examiner did not articulate any consideration of them; nor is there any evidence that the Examiner considered their specific disclosures cited in the Grounds herein for the allegedly

undisclosed limitations, nor that the examiner considered Sivasubramanian and Zha in combination. That weighs against discretionary denial. PTAB Rules for 325(d) Considerations, 89 Fed. Reg. 28693, 28705 (2024) (to be codified at 37 C.F.R. §42.108(f)) (proposed Apr. 19, 2024) (discretionary denial under §325(d) applies only if prior art was “meaningfully addressed”). Further, under *Becton, Dickinson* Factor (d), the Examiner never assessed patentability using the ***combinations*** in the Grounds (or any combination at all).

2. Part Two

If the Board reaches *Advanced Bionics* step two because Zha and a descendant of Sivasubramanian were of record, “specific teachings of [both references] impact[ing] patentability of the Challenged Claims” were “misapprehend[ed] or overlook[ed].” *Cellco P’ship v. Huawei Device Co.*, IPR2020-01117, Paper 10 at 12 (Feb. 3, 2021). The “fact that” neither reference formed a “basis of rejection weighs strongly against exercising [the Board’s] discretion to deny institution under 35 U.S.C. § 325(d).” *Id.* at 13. Furthermore, the three “*Becton, Dickson*” (“*BD*”) factors applicable to Part Two (*Advanced Bionics*, 10) weigh against denial.

Factor (c) weighs against denial because the examiner did not use any reference from the Grounds in a rejection. *Group III v. Targus*, IPR2021-00371,

Paper 21, 33 (July 9, 2021); 89 Fed. Reg. 28693, 28705 (325(d) discretionary denial only applies if art was “meaningfully addressed”).

The examiner did not identify or explain any claim limitation that either Sivasubramanian or Zha, or their combination, failed to disclose. Allowing the claims was error material to patentability because—as the Grounds explain—Sivasubramanian and Zha in combination with each other and other references render all claims obvious. Factor (e) thus weighs against denial.

The prosecution did not include the expert testimony and evidence of a POSA’s background knowledge presented in this Petition regarding why the claims are obvious. Thus, Factor (f) weighs against denial.

IX. CONCLUSION

The Board should institute review and cancel claims 1-23.

Dated: December 18, 2024

Respectfully submitted,
Tessell, Inc.

/Anant Saraswat/

Anant K. Saraswat, Reg. No. 67,336
WOLF, GREENFIELD & SACKS, P.C.

X. CLAIM LISTING

The following claim listing assigns element labels (e.g., [1A1], [1A2], etc.) to certain claims for clarity.

Claim 1
[1PRE] A method comprising:
[1A1] receiving, by a processor executing computer-readable instructions stored on a memory,
[1A2] a user request for provisioning a source database
[1A3] in a hyperconverged infrastructure system;
[1B] receiving, by the processor, selection of a database engine type;
[1C] receiving, by the processor, selection of a Service Level Agreement (“SLA”) and a protection schedule;
[1D] provisioning, by the processor, the source database based upon the database engine type; and
[1E] setting up, by the processor, automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database,
[1F] wherein the transactional logs record transactions occurring on the source database since a most recent snapshot,
[1G] and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.
Claim 2
[2PRE] The method of claim 1, further comprising:
[2A] receiving, by the processor, selection of a first set of parameters; and
[2B] creating, by the processor, a source database virtual machine based on the first set of parameters, wherein the source database resides on the source database virtual machine.
Claim 3

The method of claim 2, wherein the first set of parameters comprises at least one of a software profile, a compute profile, a network profile, and a database parameters profile.
Claim 4
[4PRE] The method of claim 1, further comprising:
[4A] converting, by the processor, a size of the source database into a number of virtual disks; and
[4B] associating, by the processor, the number of virtual disks with the source database.
Claim 5
The method of claim 1, wherein provisioning the source database comprises creating a new database as the source database.
Claim 6
The method of claim 1, wherein provisioning the source database comprises registering an existing database as the source database.
Claim 7
The method of claim 1, wherein the first frequency and the second frequency are the same.
Claim 8
[8PRE] The method of claim 1, further comprising:
[8A] presenting, by the processor, a group of database engine types that are supported by the hyperconverged infrastructure system; and
[8B] receiving, by the processor, the selection of the database engine type from the group of database engine types.
Claim 9
[9PRE] The method of claim 1, further comprising:
[9A] receiving, by the processor, selection of at least one of a sample set of parameters for creating a source database virtual machine for hosting the source database; and
[9B] creating, by the processor, the source database virtual machine based upon the at least one of the sample set of parameters.
Claim 10

[10PRE] The method of claim 1, further comprising:
[10A] presenting, by the processor, a plurality of levels of the SLA; and
[10B] receiving, by the processor, selection of one of the plurality of levels of the SLA.
Claim 11
The method of claim 1, wherein the protection schedule comprises a time period for designating at least one of a daily snapshot, a weekly snapshot, a monthly snapshot, and a quarterly snapshot.
Claim 12
[12PRE] A database management system comprising:
[12A1] a memory comprising computer-readable instructions stored thereon; and
[12A1] a processor that executes the computer-readable instructions to:
[12A3] receive a user request for provisioning a source database in a hyperconverged infrastructure system;
[12B] receive selection of a database engine type;
[12C] receive selection of a Service Level Agreement (“SLA”) and a protection schedule;
[12D] provision the source database based upon the database engine type; and
[12E] set up automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database,
[12F] wherein the transactional logs record transactions occurring on the source database since a most recent snapshot,
[12G] and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.
Claim 13
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive the user request, the selection of the database engine type, and the SLA via a dashboard.
Claim 14

The database management system of claim 12, wherein the processor further executes the computer-readable instructions to create a cloned database from the source database based on the snapshots and the transactional logs.
Claim 15
The database management system of claim 12, wherein the first frequency and the second frequency are different.
Claim 16
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive at least one parameter for creating a source database virtual machine, and associate the source database with the source database virtual machine.
Claim 17
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive a size of the source database, and convert the size into a number of virtual disks to be associated with the source database.
Claim 18
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive selection of an existing source database virtual machine, and create the source database on the existing source database virtual machine.
Claim 19
[19PRE] A non-transitory computer readable media with computer-executable instructions embodied thereon that, when executed by a processor in a hyperconverged infrastructure system, cause the processor to perform a process comprising:
[19A] receiving a user request for provisioning a source database in the hyperconverged infrastructure system;
[19B] receiving selection of a database engine type;
[19C] receiving selection of a Service Level Agreement (“SLA”) and a protection schedule;
[19D] provisioning the source database based upon the database engine type; and

[19E] setting up automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database,
[19F] wherein the transactional logs record transactions occurring on the source database since a most recent snapshot,
[19G] and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.
Claim 20
The non-transitory computer readable media of claim 19, wherein provisioning the source database comprises creating a new source database or registering an existing database as the source database.
Claim 21
[21PRE] The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for:
[21A] presenting a group of database engine types that are supported by the hyperconverged infrastructure system; and
[21B] receiving the selection of the database engine type from the group of database engine types.
Claim 22
[22PRE] The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for:
[22A] receiving selection of at least one of a sample set of parameters for creating a source database virtual machine; and
[22B] creating the source database virtual machine based upon the at least one of the sample set of parameters.
Claim 23
[23PRE] The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for:
[23A] presenting a plurality of levels of the SLA; and
[23B] receiving selection of one of the plurality of levels of the SLA.

CERTIFICATE OF SERVICE UNDER 37 C.F.R. § 42.6 (E)(4)

I certify that on December 18, 2024, I will cause a copy of the foregoing document, including any exhibits or appendices filed therewith, to be served via Priority Express Mail at the following correspondence address of record for the patent:

FOLEY & LARDNER LLP
3000 K Street N.W.
Suite 600
Washington, DC 20007-5109

Date: December 18, 2024

/MacAulay Rush/
MacAulay Rush
Paralegal
WOLF, GREENFIELD & SACKS, P.C.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 13,993 words excluding a table of contents, a table of authorities, Mandatory Notices under § 42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: December 18, 2024

/MacAulay Rush/
MacAulay Rush
Paralegal
WOLF, GREENFIELD & SACKS, P.C.