

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner

v.

KOVE IO, INC.,
Patent Owner

Patent No. 7,233,978

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,233,978**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	MANDATORY NOTICES	1
III.	PAYMENT OF FEES UNDER 37 C.F.R. §§ 42.15 AND 42.103	4
IV.	GROUND FOR STANDING.....	4
V.	PRECISE RELIEF REQUESTED AND GROUNDS RAISED.....	4
	1. Earliest Effective Filing Date of the '978 Patent.....	5
	2. Qualification as Prior Art.....	7
VI.	LEVEL OF ORDINARY SKILL	8
VII.	OVERVIEW OF THE '978 PATENT	9
VIII.	CLAIM CONSTRUCTION	11
IX.	DETAILED EXPLANATION OF GROUNDS.....	14
	A. Ground 1: Skagerwall in View of Vingralek Renders Claims 10, 12-17, 23, and 30 Obvious	14
	1. Claim 10.....	14
	2. Claim 12.....	41
	3. Claim 13	41
	4. Claim 14.....	42
	5. Claim 15	46
	6. Claim 16	47
	7. Claim 17.....	47
	8. Claim 23	61
	9. Claim 30	62
	B. Ground 2: Skagerwall in View of Vingralek and Krasner Renders Claim 11 Obvious	63
	1. Claim 11	63
	C. Ground 3: Skagerwall in View of Vingralek and Sato Renders Claims 24-26 Obvious.....	66

Petition for *Inter Partes* Review
Patent No. 7,233,978

1.	Claim 24	66
2.	Claim 25	69
3.	Claim 26	70
X.	DISCRETIONARY DENIAL IS NOT APPROPRIATE	72
A.	The Board Should Not Deny Institution Under § 325	72
B.	Discretionary Denial Under <i>General Plastic</i> Is Not Appropriate.....	74
C.	The Board Should Not Deny Institution Under <i>Fintiv</i>	75
XI.	CONCLUSION.....	77
XII.	APPENDIX A (CLAIM LISTING)	78

LIST OF EXHIBITS

Ex. 1001	U.S. Patent No. 7,233,978 to Overton et al. (“the ’978 patent”)
Ex. 1002	Declaration of Sandeep Chatterjee, Ph.D.
Ex. 1003	Curriculum Vitae of Sandeep Chatterjee, Ph.D.
Ex. 1004	Prosecution History of U.S. Patent No. 7,233,978
Ex. 1005	WIPO Application No. 1997/043717 to Kahn et al.
Ex. 1006	U.S. Patent No. 6,473,781 to Skagerwall et al.
Ex. 1007	R. Vingralek, <i>Distributed file organization with scalable cost/performance</i> (1993)
Ex. 1008	RESERVED
Ex. 1009	U.S. Patent No. 6,104,338 to Krasner
Ex. 1010	RESERVED
Ex. 1011	Japanese Patent Application Publication No. H09-231180 to Sato et al.
Exs. 1012-1014	RESERVED
Ex. 1015	M. C. Norrie et al., <i>The Database Research Group at ETH Zurich</i> , SIGMOD Record (March 1994)
Exs. 1016-1019	RESERVED
Ex. 1020	Declaration of Sylvia D. Hall-Ellis, Ph.D.
Ex. 1021	U.S. Patent Application No. 09/111,896
Ex. 1022	<i>Amazon Web Services, Inc. v. Kove IO, Inc.</i> , IPR2020-00276, Paper 23 (June 16, 2020)

Ex. 1023	Kove’s Memorandum in Opposition to Amazon Web Services, Inc.’s Motion for Judgment on the Pleadings (Dkt. Nos. 145 and 145-1) in <i>Kove IO, Inc. v. Amazon Web Services, Inc.</i> , Case No. 1:18-cv-08175 (N.D. Ill. May 29, 2020)
Ex. 1024	Memorandum Opinion and Order (Dkt. No. 484) in <i>Kove IO, Inc. v. Amazon Web Services, Inc.</i> , Case No. 1:18-cv-08175 (N.D. Ill. Dec. 17, 2021)
Ex. 1025	Memorandum Opinion and Order (Dkt. No. 739) in <i>Kove IO, Inc. v. Amazon Web Services, Inc.</i> , Case No. 1:18-cv-08175 (N.D. Ill. Feb. 6, 2024)
Exs. 1026-1032	RESERVED
Ex. 1033	2020 to 2023 Northern District of Illinois Orders Regarding Motions to Stay
Ex. 1034	Agreed Order Setting Amended Patent Case Schedule (Dkt. No. 49) in <i>Kove IO, Inc. v. Google LLC</i> , Case No. 1:23-cv-04244 (N.D. Ill. Feb. 20, 2024)
Ex. 1035	Federal Court Management Statistics (December 2023)
Ex. 1036	Memorandum by Director Vidal (June 21, 2022)
Exs. 1037-1041	RESERVED
Ex. 1042	<i>Amazon Web Services, Inc. v. Kove IO, Inc.</i> , IPR2020-00276, Paper 2 (Dec. 12, 2019)
Ex. 1043	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,233,978 in <i>Ex Parte</i> Reexamination No. 90/019,034 (Nov. 19, 2021)
Ex. 1044	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,034 (Dec. 23, 2021)
Ex. 1045	Notice of Intent to Issue <i>Ex Parte</i> Reexamination Certificate in <i>Ex Parte</i> Reexamination No. 90/019,034 (June 28, 2023)

Ex. 1046	Notice of Termination of <i>Ex Parte</i> Reexamination No. 90/019,108 (Oct. 27, 2022)
Ex. 1047	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,233,978 in <i>Ex Parte</i> Reexamination No. 90/019,109 (Aug. 31, 2022)
Ex. 1048	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,109 (Nov. 22, 2022)
Ex. 1049	Notice of Intent to Issue <i>Ex Parte</i> Reexamination Certificate in <i>Ex Parte</i> Reexamination No. 90/019,109 (Nov. 28, 2023)
Ex. 1050	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,233,978 in <i>Ex Parte</i> Reexamination No. 90/019,162 (Feb. 17, 2023)
Ex. 1051	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,162 (May 16, 2023)
Ex. 1052	Order for Staying Discovery Process until July 15, 2024 (Dkt. No. 65) in <i>Kove IO, Inc. v. Google LLC</i> , Case No. 1:23-cv-04244 (N.D. Ill. May 1, 2024)
Ex. 1053	Northern District of Illinois Local Patent Rules
Ex. 1054	Petition Decision on Request for Reconsideration in <i>Ex Parte</i> Reexamination No. 90/019,162 (June 5, 2024)
Ex. 1055	Google’s Motion to Stay and for Expedited Briefing Schedule (Dkt. No. 64) in <i>Kove IO, Inc. v. Google LLC</i> , Case No. 1:23-cv-04244 (N.D. Ill. Apr. 30, 2024)
Ex. 1056	Joseph B. Greene, <i>Oracle8 Server Unleashed</i> (1998)
Ex. 1057	Joseph B. Greene, <i>Oracle DBA Survival Guide</i> (1995)
Ex. 1058	Betty Salzberg, <i>File Structures</i> (1988)
Ex. 1059	Andrew P. Black & Yeshayahu Artsy, <i>Implementing Location Independent Invocation</i> (1989)

I. INTRODUCTION

Google LLC (“Petitioner”) respectfully requests *inter partes* review of claims 10-17, 23-26, and 30 (“the challenged claims”) of U.S. Patent No. 7,233,978 (“the ’978 patent”) (Ex. 1001) assigned to Kove IO, Inc. (“Patent Owner”).

II. MANDATORY NOTICES

Real Parties-in-Interest: Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner identifies Google LLC as the real party-in-interest.¹

Related Matters: The ’978 patent is asserted in *Kove IO, Inc. v. Amazon Web Services, Inc.*, Case No. 1:18-cv-08175 (N.D. Ill.) (“the AWS Proceeding”), and *Kove IO, Inc. v. Google LLC*, Case No. 1:23-cv-04244 (N.D. Ill.).

The ’978 patent was at issue in *Amazon Web Services, Inc. v. Kove IO, Inc.*, IPR2020-00276 (“the AWS IPR”). Amazon Web Services (“AWS”) filed the AWS IPR (Ex. 1042) in December 2019 presenting grounds based on different combinations of prior art from the grounds here (*id.*, 23-24; *infra* §§IV, IX), and

¹ Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

institution was denied (Ex. 1022, 24). Petitioner had no involvement in the preparation or filing of the AWS IPR.

The '978 patent was also at issue in ex parte reexamination Nos. 90/019,034 ("the '034 EPR"), 90/019,108 ("the '108 EPR"), and 90/019,109 ("the '109 EPR"), and is currently pending in ex parte reexamination No. 90/019,162 ("the '162 EPR").

The '034 EPR was filed by third party AWS (Ex. 1043) in November 2021 presenting SNQs on claims 1, 3, 6, 10, 14, 17, 23, 24, 30, and 31 based on different combinations of prior art from the grounds here (*id.*, 12-14; *infra* §§IV, IX); the request was granted (Ex. 1044, 18), with claims 1 and 31 being canceled and the rest of the claims confirmed (Ex. 1045, 5). Petitioner had no involvement in the preparation or filing of the '034 EPR.

The '108 EPR was filed by third party AWS in August 2022 but was terminated for failing to comply with filing requirements. (Ex. 1046, 3.) Petitioner had no involvement in the preparation or filing of the '108 EPR.

The '109 EPR was filed by third party AWS (Ex. 1047) in August 2022, presenting SNQs on claims 17, 23, 24, and 30 based on different combinations of prior art from the grounds here (*id.*, 15-18; *infra* §§IV, IX). The request was granted with respect to all of claims 17-30, with the examiner including claims 18-22 and 25-29 *sua sponte* because they all depend from claim 17. (Ex. 1048, 39-

40.) The claims were confirmed. (Ex. 1049, 8.) Petitioner had no involvement in the preparation or filing of the '109 EPR.

The '162 EPR was filed by third party AWS (Ex. 1050) in February 2023, presenting SNQs on claims 3, 6, 10, and 14 based on different combinations of prior art from the grounds here (*id.*, 21-24; *infra* §§IV, IX.). Petitioner had no involvement in the preparation or filing of the '162 EPR. Because the request raised substantial new questions of patentability, reexamination was ordered on claim 6 in May 2023 (Ex. 1051, 52), and then on all remaining claims 3, 10, and 14 in June 2024 after a decision on AWS's petition for review (Ex. 1054, 9). The examiner has not issued an Office Action at the time of the filing of the current Petition.

Petitioner is concurrently filing IPR petitions challenging related U.S. Patent Nos. 7,103,640 ("the '640 patent") and 7,814,170 ("the '170 patent"), which are also at issue in the above-noted district court cases.

Counsel and Service Information: Lead counsel is Naveen Modi (Reg. No. 46,224). Backup counsel are Joseph E. Palys (Reg. No. 46,508), Daniel Zeilberger (Reg. No. 65,349), Howard Herr (*pro hac vice* to be filed), Alexa Lowman (*pro hac vice* to be filed), and Puja Patel (*pro hac vice* to be filed). Service information is Paul Hastings LLP, 2050 M Street NW, Washington, D.C.

20036, Tel.: 202.551.1700, Fax: 202.551.1705, email: PH-Google-Kove-IPR@paulhastings.com. Petitioner consents to electronic service.

III. PAYMENT OF FEES UNDER 37 C.F.R. §§ 42.15 AND 42.103

The PTO is authorized to charge any fees due during this proceeding to Deposit Account No. 50-2613.

IV. GROUNDS FOR STANDING

Petitioner certifies under 37 C.F.R. § 42.104(a) the '978 patent is available for review and Petitioner is not barred or estopped from requesting review on the grounds identified herein.

V. PRECISE RELIEF REQUESTED AND GROUNDS RAISED

Petitioner requests review of claims 10-17, 23-26, and 30 of the '978 patent and cancellation of these claims as unpatentable on the following grounds:

Ground 1: Claims 10, 12-17, 23, and 30 are unpatentable under pre-AIA 35 U.S.C. § 103 as being obvious over Skagerwall (Ex. 1006) and Vingralek (Ex. 1007).

Ground 2: Claim 11 is unpatentable under pre-AIA 35 U.S.C. § 103 as being obvious over Skagerwall, Vingralek, and Krasner (Ex. 1009).

Ground 3: Claims 24-26 are unpatentable under pre-AIA 35 U.S.C. § 103 as being obvious over Skagerwall, Vingralek, and Sato (Ex. 1011).

As discussed below, Skagerwall, Vingralek, Krasner, and Sato qualify as prior art and are properly relied upon for showing unpatentability of the '978 patent.

1. Earliest Effective Filing Date of the '978 Patent

The '978 patent issued from U.S. Patent Application No. 09/872,736, filed on June 1, 2001, which claims priority to provisional application No. 60/277,408 filed on March 19, 2001; application No. 09/661,222 (now the '640 patent) filed on September 13, 2000; provisional application No. 60/209,070 filed on June 2, 2000; application No. 09/503,441 filed on February 14, 2000; application No. 09/367,461 filed on August 13, 1999; and application No. 09/111,896 ("the '896 application") filed on July 8, 1998. (Ex. 1001, Cover, 1:9-26.) However, challenged claims 10-17, 23-26, and 30 of the '978 patent are not entitled to claim priority to the '896 application because those claims are not fully supported by the '896 application. (Ex. 1002, ¶¶32-36.) Therefore, the earliest possible effective filing date for the challenged claims of the '978 patent is August 13, 1999.²

In order for a claim to be entitled to the benefit of the filing date of an earlier filed application, the subject matter of the claim must be disclosed in the earlier-

² Petitioner does not concede that the '978 patent is entitled to an August 13, 1999 priority date.

filed application in accordance with the written description requirement of Section 112. *PowerOasis, Inc. v. T-Mobile USA, Inc.*, 522 F.3d 1299, 1306 (Fed. Cir. 2008). Here, none of the challenged independent claims of the '978 patent (i.e., claims 10, 14, and 17) are disclosed in the '896 application; therefore, none of the challenged claims of the '978 patent are entitled to claim priority to the '896 application.³

The '896 application does not provide written description support for at least the following limitations of the '978 patent: limitations [10.b], [14.c], [17.pre], and [17.e]. (Ex. 1002, ¶¶32-36.) Limitations [10.b] and [14.c] require that if a location server does not contain the requested location information associated with the desired identifier or entity in the request, the server returns a redirect message with information for finding a location server that does contain the requested location information. The '896 application does not provide support for any such redirect message, nor does it provide support for any of the operations associated with a redirect message. At most, the '896 application describes a “server response message[]” that “answers client queries, and sometimes also performs actions on

³ Patent Owner acknowledged in the AWS Proceeding that of all claims in the '978 patent and the related '640 and '170 patents, only claims 1 and 31 of the '978 patent are allegedly supported by the '896 application. (Ex. 1023, 7 (n.4).)

behalf of the client called server actions.” (Ex. 1021, 15:23-26.) Although the ’896 application discloses server response messages may be sent in response to a number of events, it does not disclose the claimed “redirect message” and associated operations recited in claims 10 and 14, as described above. (*Id.*, 15:27-24:24, 29:7-41:10.) Limitations [17.pre] and [17.e] require a method of scaling capacity or transaction rate capability in a location server, where a portion of identifiers and associated locations are transferred from a first to second location server when a performance criterion of the first location server is reached. The ’896 application does not provide support for any method of scaling capacity or transaction rate capability in a location server, nor providing support for transferring identifiers and locations to another location server when a performance criterion is reached. At most, the ’896 application describes the system is scalable. (*Id.*, 10:3-6, 11:21-12:3, 13:13-15.) The examiner in the ’162 EPR likewise found the “claims [3, 6, 10, and 14]...are not entitled to the priority date of the [’896 application].” (Ex. 1051, 20.)

2. Qualification as Prior Art

Skagerwall issued from an application filed October 27, 1998 (Ex. 1006, Cover), and thus qualifies as prior art at least under pre-AIA 35 U.S.C. § 102(e).

Vingralek is a journal article published by ETH Zürich as early as December 1993, and thus is prior art under at least pre-AIA 35 U.S.C. § 102(b). The

declaration of Dr. Sylvia D. Hall-Ellis (Ex. 1020, ¶¶1-66) confirms that Vingralek was published, disseminated, and reasonably available to the public on or shortly after December 31, 1993. (Ex. 1020, ¶¶46-54; *see also id.*, ¶¶19-45.) For example, Dr. Hall-Ellis explains that based on finding a copy of Vingralek that is dated December 1993 in the ETH Zürich Library and a MARC record in its online library catalog indicating that the ETH Zurich journal was first published in 1974. (*Id.*, ¶¶51-54.) Moreover, Vingralek was cited by Ex. 1015 (“Norrie”), which was published on March 1, 1994. (*Id.*, ¶¶55-63.) It is her opinion that Vingralek was publicly available on or shortly after December 31, 1993, as shown on the cover. (*Id.*, ¶¶64-65.)

Krasner issued from an application filed May 14, 1998 (Ex. 1009, Cover), and thus qualifies as prior art at least under pre-AIA 35 U.S.C. § 102(e).

Sato is a publication of a Japanese patent application which was published September 5, 1997, and thus qualifies as prior art at least under pre-AIA 35 U.S.C. § 102(a).

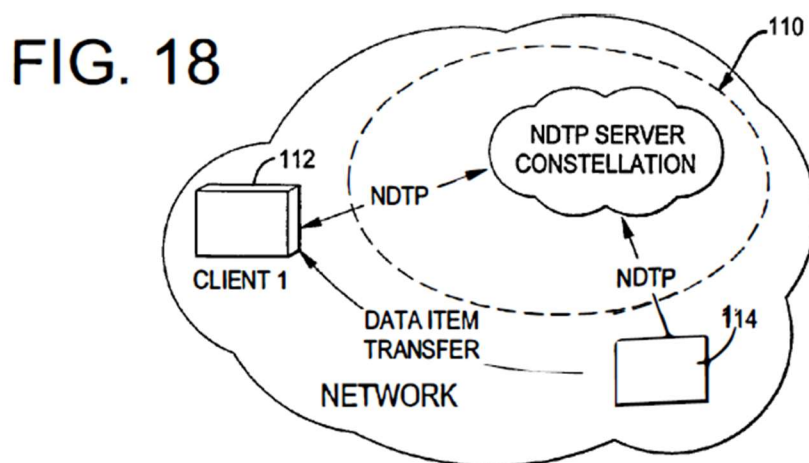
VI. LEVEL OF ORDINARY SKILL

A person of ordinary skill in the art at the time of the alleged invention of the '978 patent (“POSITA”) would have had an undergraduate degree in electrical engineering, computer engineering, computer science or a related field along with two years of work experience in the field of distributed computing systems. (Ex.

1002, ¶¶20-21.)⁴ More education can supplement practical experience and vice versa. (*Id.*)

VII. OVERVIEW OF THE '978 PATENT

The '978 patent describes storing and retrieving information across a network (Ex. 1001, 1:30-33, 1:56-58) and a network distributed tracking protocol (NDTP) (*id.*, 2:19-21, 7:55-61). Figure 18 shows an NDTP server constellation where a client consults servers that return strings in response to a client request. (*Id.*, 18:50-58.)



(*Id.*, FIG. 18; Ex. 1002, ¶29.)

Server constellations have two organizational approaches: client-centric (Figure 19) and server-centric (Figure 20). In the client-centric approach,

⁴ Petitioner submits the declaration of Dr. Sandeep Chatterjee (Ex. 1002), an expert in the field of the '978 patent. (Ex. 1002, ¶¶1-28; Ex. 1003.)

processing emphasis is toward clients (*id.*, 18:60-19:3), whereas in the server-centric approach, server-processing is emphasized (*id.*, 19:20-24.)

FIG. 19

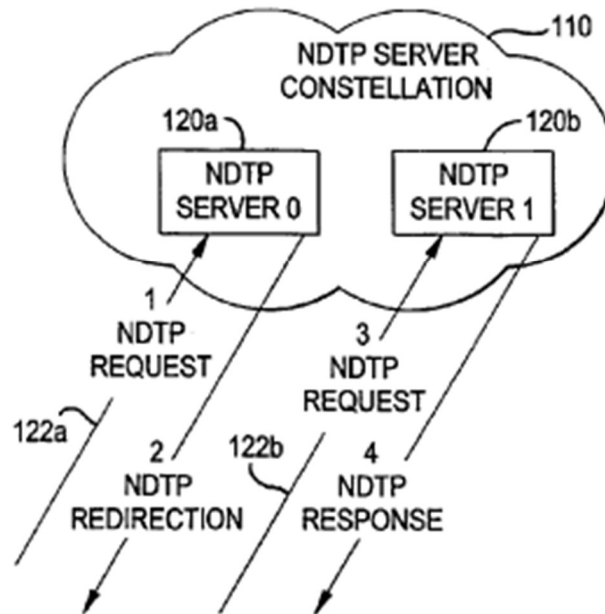
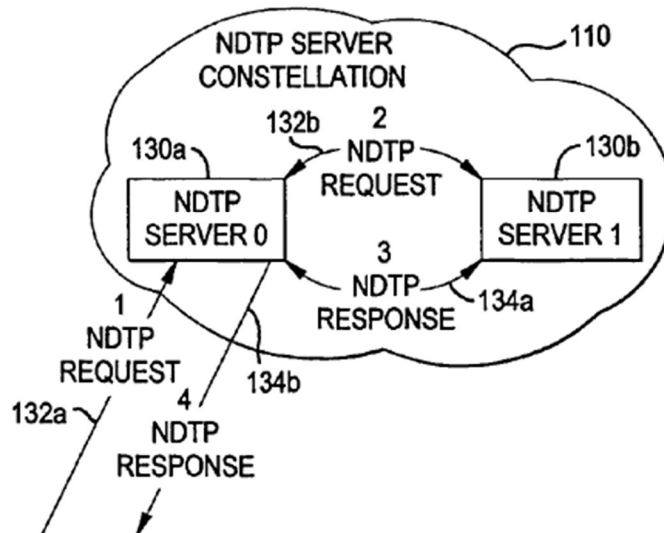


FIG. 20



(*Id.*, FIGS. 19, 20; Ex. 1002, ¶30.)

In both approaches, a client transmits an identifier specifying an entity to a server in a client request. (Ex. 1001, 2:19-52.) If the server contains location information specifying network locations of data associated with the entity, the server returns the location information. (*Id.*; Ex. 1002, ¶31.)

As demonstrated below, all of the features recited in the challenged claims of the '978 patent were known in the prior art before the alleged invention date of the '978 patent. (*Infra* §IX; Ex. 1002, ¶¶22-28, 37-49; Exs. 1005, 1011, 1056-1059.)

VIII. CLAIM CONSTRUCTION

The Board only construes the claims when necessary to resolve the underlying controversy. *Toyota Motor Corp. v. Cellport Sys., Inc.*, IPR2015-00633, Paper No. 11 at 16 (Aug. 14, 2015). For purposes of this proceeding, Petitioner believes no special constructions are necessary to assess whether the challenged claims are unpatentable over the asserted prior art. (Ex. 1002, ¶¶50-56.)⁵

⁵ Petitioner reserves all rights to raise claim construction and other arguments in other venues.

As shown in the table below, in the AWS IPR, the Board construed a claim term of the '978 patent (Ex. 1022, 13), and in the AWS Proceeding, the district construed several claim terms of the '978 patent. (Ex. 1024, 35; Ex. 1025, 19.) While Petitioner does not acquiesce to those constructions, the prior art mappings found in §IX of this Petition explain how the claims of the '978 patent are disclosed and/or rendered obvious if those constructions are applied. (Ex. 1002, ¶¶50-56.)

'978 Patent Terms	AWS Proceeding/IPR Construction
"identifier"	"a unique encoding that identifies an individual entity, and with which zero or more data location specifiers are associated in a location server" (Ex. 1022, 13.)
"location information"	"one or more identifiers and their associated locations" (Ex. 1024, 35.)
"location server"	"a network-attached component that maintains a set of identifier/location mappings that are modified or returned

'978 Patent Terms	AWS Proceeding/IPR Construction
	in response to location request messages from clients” (Ex. 1024, 35.)
“client”	“a network-attached component (which may be software or hardware) that initiates update or lookup of identifier/location mappings from a location server with location request messages” (Ex. 1024, 35.)
“plurality of location servers”	“location servers in non-hierarchical structures” (Ex. 1025, 19.)

IX. DETAILED EXPLANATION OF GROUNDS

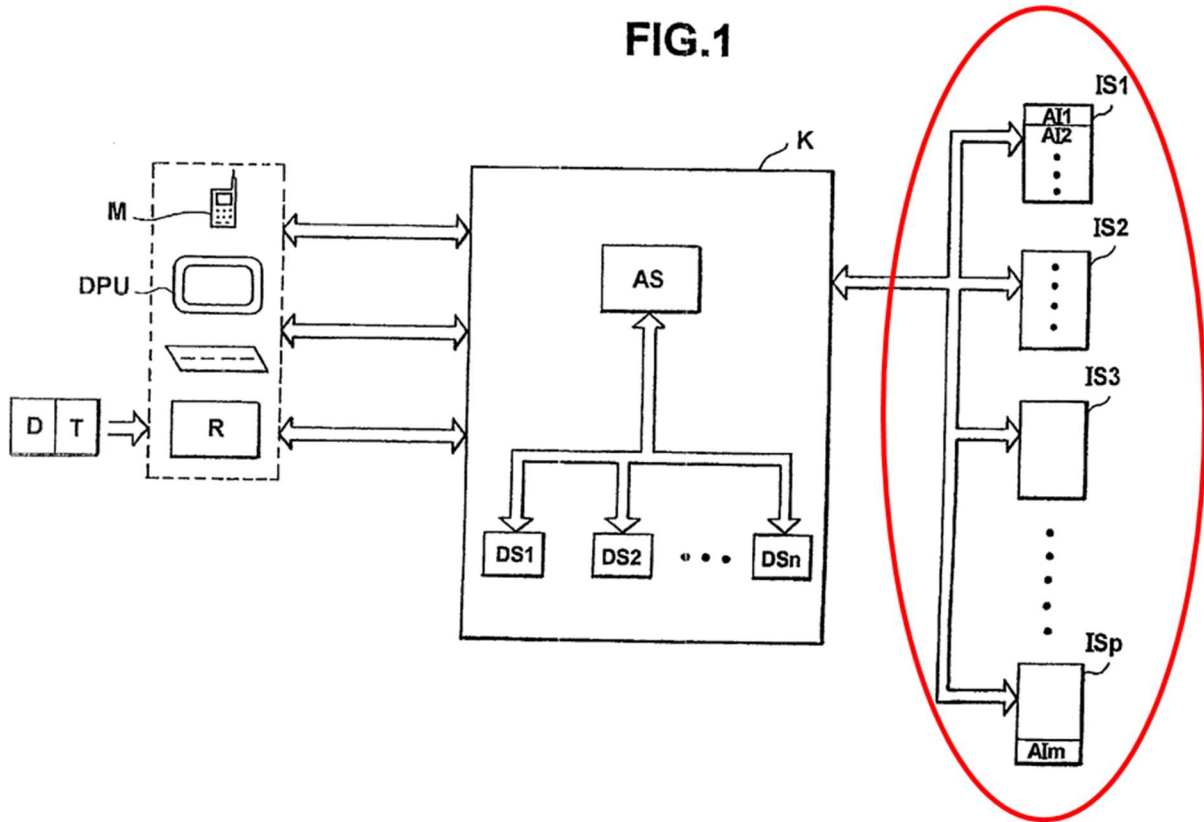
As detailed below, each of the challenged claims are unpatentable.

A. Ground 1: Skagerwall in View of Vingralek Renders Claims 10, 12-17, 23, and 30 Obvious

1. Claim 10

i) [10.pre]

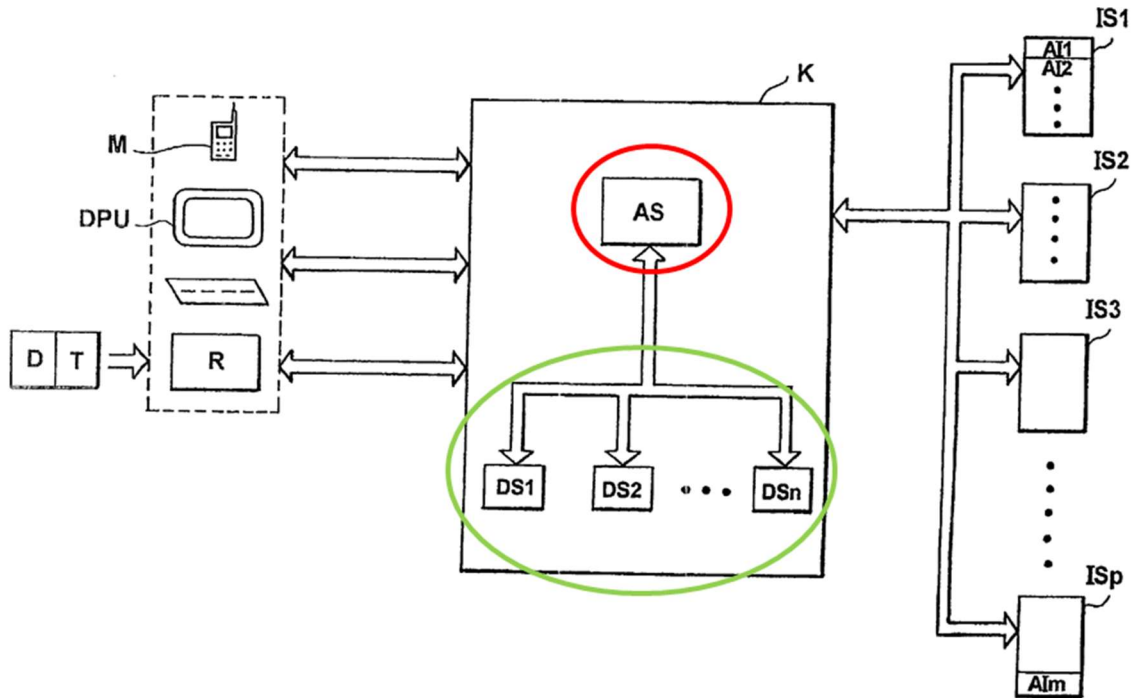
To the extent the preamble of claim 10 is limiting, Skagerwall discloses the limitations therein. (Ex. 1002, ¶¶58-67.) Skagerwall discloses a system that links objects, e.g., products in the physical world, to different types of information related to the objects. (Ex. 1006, Abstract (“linking each of...tagged object[s] to information services stored on a network”), 6:45-48.) The system links objects to information or application data packets about those objects. (*Id.*, 4:1-5, 4:6-11 (“Each...application data packet[] may be associated with one of the...objects of the real world, i.e., it contains information related to the object.”).) The application data packets may include text, video, or programs. (*Id.*, 4:5-9, 7:13-17.) Skagerwall discloses with respect to Figure 1 the application data packets, each identified by an application identifier (AI1-AIm), are distributed across application servers on a network, each identified by an address (IS1-ISp). (*Id.*, 6:34-37, 7:4-8.)



(*Id.*, FIG. 1; Ex. 1002, ¶58.)

The system includes “a plurality of location servers for managing location information.” Skagerwall’s system includes an access server (red) and a network of directory servers (green) (“a plurality of location servers”). (Ex. 1006, 6:32-34.)

FIG.1



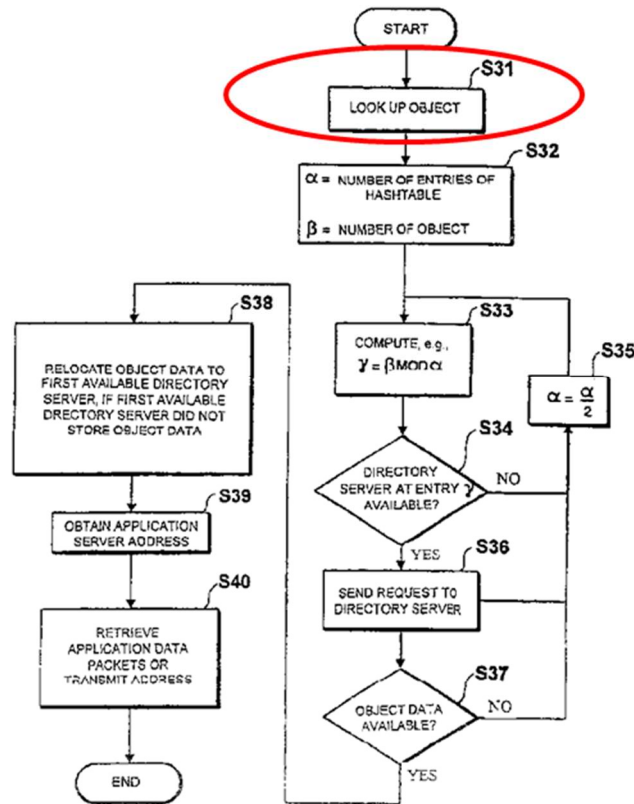
(*Id.*, FIG. 1.) Each directory server (“location server”) stores object data, which associates objects with application data packets, and is involved in the retrieval of application data packets from application servers. (*Id.*, Abstract, 4:13-16, 8:19-22, FIG. 1.) The object data includes object tags, associated application identifiers of application data packets, and addresses of the application servers storing those packets (collectively “location information”), and provides information about where an application data packet is stored. (*Id.*, 8:22-27, 15:1-52, 19:35-39, 19:51-53.) A POSITA would have understood the directory servers (“plurality of location servers”) manage the object tags, application identifiers, and application

server addresses (“for managing location information”) since they are responsible for storing such information. (Ex. 1002, ¶59.)

Skagerwall also discloses the “plurality of location servers” are for “providing location information to location queries.” Skagerwall discloses a two-step process (described below) for retrieving application data packets, where the first step comprises identifying a directory server storing object data relating to an object and retrieving that object data, and the second step involves using the object data to retrieve application data packets from the application servers. (Ex. 1006, 8:22-36; Ex. 1002, ¶60.)

The system kernel includes an access server that receives an information request about an object (step S31 of Figure 3 below), where the request at least includes an object tag. (*Id.*, 6:41-43, 9:1-5, 12:27-32, 13:23-26, 13:29-30, FIG. 2a, FIG. 3.)

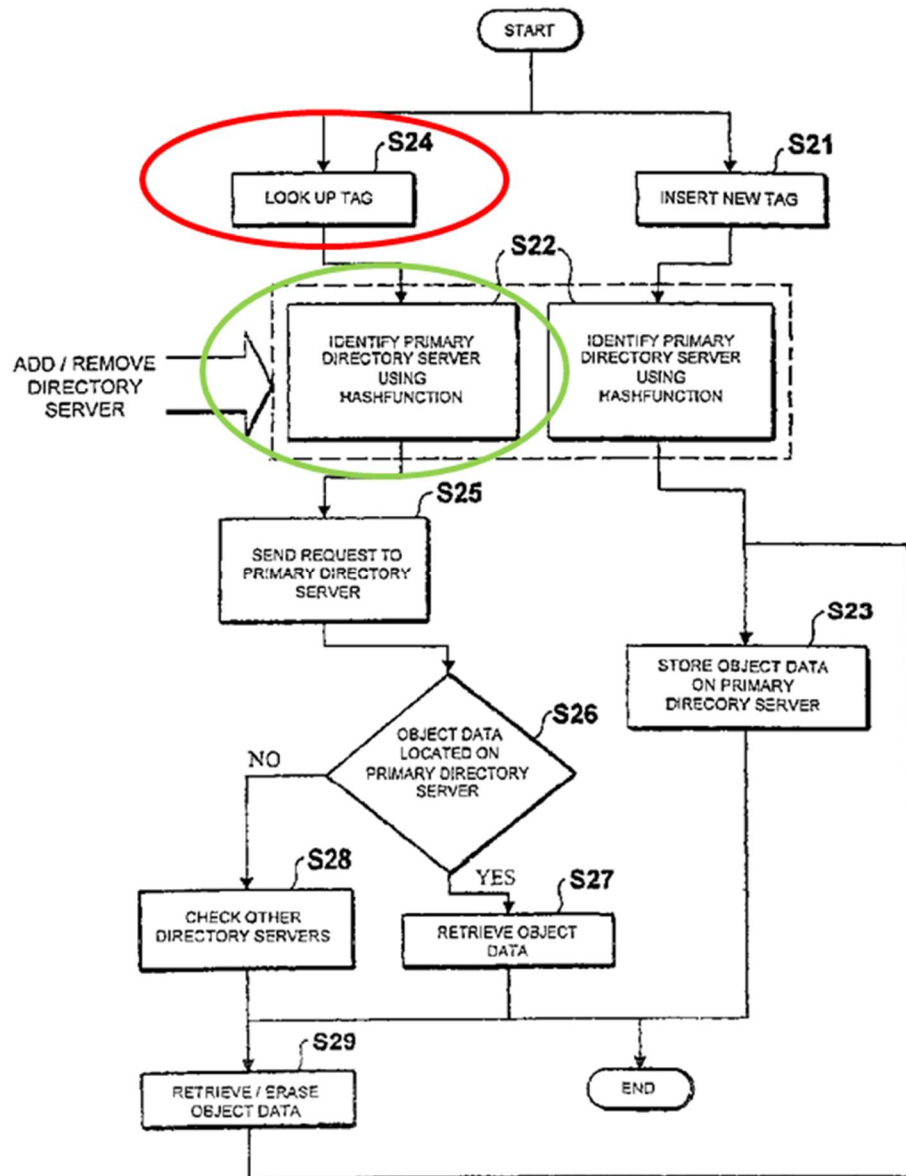
FIG.3



(*Id.*, FIG. 3; Ex. 1002, ¶61.)

Upon receiving the information request, the access server identifies the directory server (“location server”) storing object data concerning the location of application data packets. (Ex. 1006, 9:18-22.) As shown in Figure 2a, after receiving an information request including an object tag in step S24 (red), a primary directory server (“location server”) is determined in step S22 (green) using a hash function applied to the object tag, with the result serving as an index to a table listing available directory servers. (*Id.*, 12:8-12, 12:30-37; *see also id.*, 10:28-11:16, 13:15-57; Ex. 1002, ¶62.)

FIG.2a

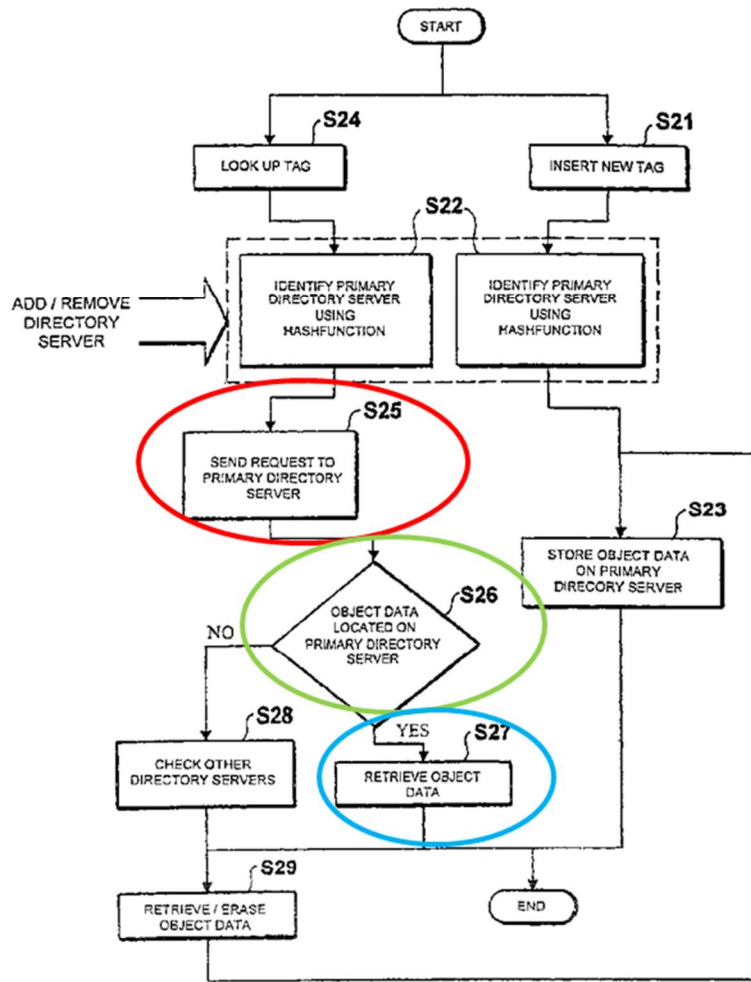


(Ex. 1006, FIG. 2a.)

In step S25 (red in Figure 2a below), the access server then sends a request (“location quer[y]”) including the object tag to the first directory server (“location server”) to retrieve object tags, application server addresses, and application

identifiers (“providing location information to location queries”). (*Id.*, 9:9-13, 9:23-25, 19:42-46, 19:51-55; *see also id.*, 9:42-48, 14:7-11, 12:37-40, FIG. 2a (step S25).) In step S26 (green), the directory server (“location server”) determines whether it contains the object data associated with the object tag. (*Id.*, 8:22-27, 12:41-43.) If the directory server contains the object data, then in step S27 (blue), that object data (object tag, application server addresses, and application identifiers) is retrieved and subsequent steps for providing the user with the desired information are performed (e.g., retrieving application data packets from application servers based on the object data). (*Id.*, 12:54-62; Ex. 1002, ¶63.)

FIG.2a



(Ex. 1006, FIG. 2a.)

Skagerwall also discloses this limitation as construed in the AWS Proceeding. Skagerwall discloses the claimed “location server[]” under the AWS Proceeding’s construction. (§VIII; Ex. 1001, 4:34-38.) Skagerwall discloses that each directory server is a component attached to a distributed network (i.e., a network-attached component). (Ex. 1006, 8:15-19 (“the data base of object data is distributed over a plurality of directory servers”), 16:34-36 (all components in the

system may be “connected via a network N for communication with each other”), FIG. 1 (depicting a network of directory servers DS; DS1-DSn).) As discussed for limitation [10.b], each object tag is a claimed “identifier” under the AWS Proceeding construction. (§IX.A.1.iii.) Skagerwall’s application identifiers and application server addresses both disclose “location[s]” (or similar terms) as described in the ’978 patent. (Ex. 1001, 4:28-30; Ex. 1006, 8:22-27, 15:1-52.) A POSITA would have also understood that each directory server maintains mappings from object tags (i.e., identifiers) to application server addresses and application identifiers (i.e., locations of application servers storing application data packets) because Skagerwall discloses that object tags and associated object data are distributed across directory servers based on a hash function, and that each directory server has a memory means associating object tags with corresponding object data (including application server addresses and application identifiers). (Ex. 1002, ¶64; Ex. 1006, 8:43-63, 15:15-20, 15:49-52.)

Further, Skagerwall discloses that the directory server returns identifier/location mappings in response to a request from the access server for location information associated with an object tag (i.e., in response to location request messages from a client). For example, as discussed above and for limitation [10.b], the object tag, application server addresses, and application identifiers (i.e., identifier/location mappings) are returned to the access server if the

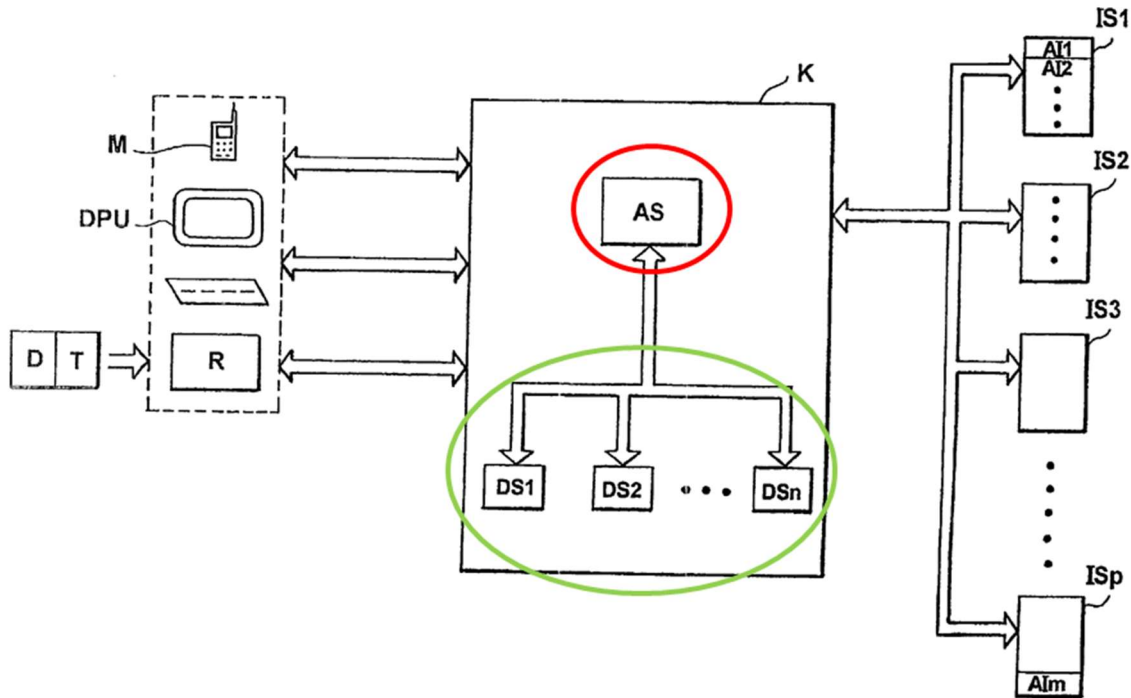
directory server determines it contains the requested location information.

(§IX.A.1.iii; Ex. 1006, 19:42-46, 19:51-55; Ex. 1001, 2:32-37, 4:61-64.)

Skagerwall further discloses that identifier/location mappings may be updated/modified in response to a request from the access server to add new location information (i.e., another type of location request message from the client). (Ex. 1006, 12:3-26, 14:29-56; Ex. 1001, 10:1-4.) Thus, Skagerwall discloses the claimed “location server[]” under the AWS Proceeding’s construction. (Ex. 1002, ¶65.)

Further, Skagerwall discloses the claimed “plurality of location servers” under the AWS Proceeding’s construction. (§VIII.) A POSITA would have understood the directory servers (each a “location server”) are arranged in a flat, cluster topology, rather than a hierarchical or tree-like structure where servers can only communicate with nodes immediately above or below it in the tree, because—as shown below in Figure 1—there is no hierarchy of intermediary servers between the access server (in red) and the directory servers (in green), and the directory servers themselves are not arranged in a hierarchy. (Ex. 1002, ¶66.)

FIG.1



(Ex. 1006, FIG. 1.) Further, Skagerwall discloses that the hash function is used to distribute object tags and associated object data (e.g., application server addresses) among the directory servers. (*Id.*, 8:15-27, 8:38-54, 8:55-59.) A POSITA would have thus understood that because a hash function is used for distribution across directory servers, Skagerwall’s directory servers are arranged in a non-hierarchical configuration. (Ex. 1002, ¶66.) Indeed, consistent with the above, Patent Owner argued in the AWS Proceeding that the hash function feature requires a non-hierarchical server structure. (Ex. 1025, 15.)

Skagerwall also discloses the claimed “location information” under the AWS Proceeding’s construction. (§VIII.) As discussed above, the object tags

stored on a directory server are identifiers of objects, and the application server addresses and application identifiers are both locations associated with an object tag. (Ex. 1002, ¶67.)

ii) [10.a]

Skagerwall discloses and/or suggests this limitation. (Ex. 1002, ¶¶68-72.)

First, Skagerwall discloses “a plurality of location servers containing location information corresponding to a plurality of entities.” As discussed for limitation [10.pre], Skagerwall discloses a network of directory servers (“a plurality of location servers”) that contain object tags, associated application identifiers, and associated application server addresses (collectively “location information”) corresponding to multiple objects (“corresponding to a plurality of entities”). (§IX.A.1.i; Ex. 1006, 8:15-27.)

Second, Skagerwall discloses and/or suggests “the location information formatted according to a transfer protocol configured for manipulating location information.” A POSITA would have found it beneficial to format communications sent between an access server and directory server according to a transfer protocol, e.g., hypertext transfer protocol (HTTP). Skagerwall discloses that its system may “use standard and/or amended protocols of a known datacom- or telecom network for retrieving information associated with an object.” (Ex. 1006, 9:59-62.) Skagerwall also discloses that the system may be “used in

combination with a known datacom- or telecom network, such as the Internet,” and that at least lookup procedures may be performed “via a HTTP request.” (*Id.*, 18:2-7, 18:24-26.) A POSITA would have understood HTTP is a transfer protocol broadly governing communications between web clients and web servers, and its application is not limited to communications occurring during lookup procedures. Moreover, a POSITA would have found it beneficial to use a transfer protocol compatible with Skagerwall’s system, e.g., HTTP, for not only the lookup processes, but also other processes involving the manipulation (e.g., storage, retrieval, etc.) of object data to enable the system to operate using a widely-used, Internet-compatible, and well-known protocol to improve reliability. Thus, a POSITA would have found it obvious to format communications (e.g., requests) between an access server and directory server, and the storage/retrieval of the location information on a directory server (“manipulating location information”), in accordance with HTTP (“the location information formatted according to a transfer protocol configured for manipulating location information”). (Ex. 1002, ¶69.)

A POSITA would have had the skill and reasonable expectation of success in implementing this feature because it would have involved modifying the system in a manner Skagerwall suggests is feasible—implementing the system in accordance with a standard protocol, e.g., HTTP. (Ex. 1002, ¶70.) Moreover, the

modification would have involved applications of known technologies (e.g., known communication systems of Skagerwall) according to known methods (e.g., known transfer protocol used in Skagerwall or HTTP) to yield the predictable result of a system having communications and processes for the manipulation of the disclosed “location information” formatted according to HTTP, as discussed above. *KSR Int’l. Co. v. Teleflex Inc.*, 550 U.S. 398, 416 (2007).

Third, Skagerwall discloses “the location information...comprising at least one application server address.” (Ex. 1002, ¶71.) As discussed for limitation [10.pre], the “location information” includes application server addresses (“at least one application server address”). (§IX.A.1.i.)

Fourth, Skagerwall discloses “the plurality of location servers are arranged in a cluster topology such that each location server contains a unique set of location information of an aggregate set of the location information.” As discussed for limitation [10.pre], a POSITA would have understood the directory servers (“the plurality of location servers”) are arranged in a cluster structure (“arranged in a cluster topology”). (§IX.A.1.i.) Skagerwall further discloses the database of object tags and associated object data (including application server addresses and application identifiers) is distributed evenly across multiple directory servers, such that each directory server stores object data related to certain object tags. (Ex. 1006, 8:15-27, 8:48-51.) A POSITA would have thus understood each directory

server (“location server”) contains a unique set of object tags, application server addresses, and application identifiers (collectively “location information”) out of the full set of all the object tags, application server addresses, and application identifiers (“an aggregate set of the location information”). (Ex. 1002, ¶72.)

iii) [10.b]

Skagerwall in view of Vingralek discloses and/or suggests this limitation. (Ex. 1002, ¶¶73-91.) **First**, Skagerwall discloses “programming logic stored on each of the plurality of location servers” that is configured to perform operations performed by a “location server” discussed herein. Skagerwall discloses each directory server (“each of the plurality of location servers”) has a central processing unit that controls the operations of the server. (Ex. 1006, 15:1-7 (“The directory server DS1 comprises a central processing unit CPU for controlling operations of the directory server.”).) Thus, each directory server would store programming logic (“programming logic stored on each of the plurality of location servers”) to perform operations thereof, including those recited in limitation [10.b] and discussed below. (Ex. 1002, ¶73.)

Second, Skagerwall discloses “programming logic...responsive to a location query for a desired identifier to return one of a location message, wherein a queried location server returns a location message if the queried location server contains location information for the desired identifier.” As discussed for limitation

[10.pre], the primary directory server (“queried location server”) receives a request (“location query”) from the access server for information concerning the location of application data packets related to an object tag (“for a desired identifier”), where that request includes the corresponding object tag. (§IX.A.1.i.) The directory server then determines whether it stores the mapping of an object tag to addresses of application servers and application identifiers (the three types of information collectively “location information”) for the object tag included in the request (“for a desired identifier”). (*Id.*) If the directory server (“queried location server”) contains the disclosed “location information” associated with the object tag (“for a desired identifier”), the disclosed “location information” is retrieved and subsequent steps for providing the user with the desired information are performed. (*Id.*) For example, as discussed for limitation [10.pre], Skagerwall discloses the disclosed “location information” may be sent to the access server (“returns a location message”), which the access server may use to determine where to find the application data packets associated with an object tag. (*Id.*) A POSITA would have understood the “location information” is returned in a message because Skagerwall discloses that other communications from directory server to access server are “message[s].” (Ex. 1006, 13:5-6, 13:64-65; Ex. 1002, ¶74.)

The object tag is an “identifier” under the AWS IPR construction. (§VIII; Ex. 1001, 4:25-28.) A POSITA would have understood the object tag is an

encoding that uniquely identifies an object (i.e., an individual entity) because, as discussed for limitation [10.pre], the tag is included in a query to identify the specific object for which information is sought. (§IX.A.1.i; Ex. 1006, 3:4-5, 7:21-25, 16:5-9.) As further discussed, zero or more application identifiers and application server addresses (both data location specifiers), which specify the location of application data packets (i.e., data) are associated with the object tag in a directory server (i.e., a location server, as construed in the AWS Proceeding). (§IX.A.1.i; Ex. 1006, 8:15-27, 15:4-52; Ex. 1002, ¶75.)

Third, Skagerwall discloses “if the queried location server does not contain location information relevant to the desired identifier,” the directory server (“queried location server”) is configured to generate “information for finding a location server having location information related to the desired identifier.” (Ex. 1002, ¶76.) As discussed for limitation [10.pre] with respect to Figure 2a, a primary directory server is determined in step S22 based on the application of a hash function to an object tag, with the result used as an index for searching the hash table. (§IX.A.1.i.) As further discussed therein, in step S26 the primary directory server (“queried location server”) determines whether it stores the mapping of an object tag to application server addresses and application identifiers (the three types of information collectively “location information”) for the object tag included in the request (“relevant to the desired identifier”). (*Id.*) If the

queried server includes the requested information, then the disclosed “location information” is retrieved by the access server in step S27. (*Id.*) If the directory server determines in step S26 that it does not contain the requested information (“if the queried location server does not contain location information relevant to the desired identifier”), then the request is forwarded to another directory server in step S28, which is determined based on the application of the hash function. (Ex. 1006, 12:63-13:5.)

Skagerwall discloses the primary directory server (“queried location server”) may be the component that determines the location of a “correct” directory server (i.e., the directory server that has the requested application server addresses and application identifiers associated with the object tag) using the hash function. (*Id.*, 9:18-28, 10:22-25; Ex. 1002, ¶77.)

Thus, Skagerwall discloses that “if the queried location server does not contain location information relevant to the desired identifier,” the queried directory server (“queried location server”) is configured to generate “information for finding a location server having location information related to the desired identifier.” (Ex. 1002, ¶78.)

To the extent that Skagerwall does not explicitly disclose “**programming logic...responsive to a location query for a desired identifier to return...a redirect message** if the queried location server does not contain location

information relevant to the desired identifier, **wherein the redirect message comprises information for finding a location server having location information related to the desired identifier,**” a POSITA would have been motivated and found it obvious to implement this feature in view of Vingralek. (Ex. 1002, ¶79.)

Like Skagerwall, Vingralek relates to storing data in a distributed network. Vingralek discloses a “distributed file organization system” where a file is “organized into buckets that are spread across multiple servers” (Ex. 1007, 4) and clients can modify the file’s distribution by issuing requests to insert/delete/retrieve records for a given key *K* (akin to an identifier of a record) in the file, such that the file grows or shrinks dynamically (*id.*, 9). Vingralek offers “a new approach for hash-based distributed files that strives for...‘cost-conscious’ scalability...to minimize the number of servers...used for providing the necessary performance capacity.” (*Id.*, 6.) Clients locate servers for insertion/update/deletion, and perform “key-based retrieval through a hash function whose target domain is dynamically expanded as buckets are split and servers are added, based on the principle of linear hashing.” (*Id.*, 5, 8.) Vingralek’s hash-based distributed file system is implemented using an address table, which a POSITA would have understood is akin to a hash table, to look up the server that corresponds to a hashed key. (*Id.*, 4, 6.) Thus, Vingralek is similar to Skagerwall, which discloses

that the database of object data is distributed based on a hash function over a plurality of directory servers, so that the number of objects may be scaled by adding new directory servers. (Ex. 1006, 8:15-19, 8:43-54; §IX.A.1.i.) A POSITA would have therefore been motivated to consider the teachings of Vingralek when implementing Skagerwall's system/method. (Ex. 1002, ¶80.)

Vingralek additionally discloses propagating updates to clients' and other servers' address tables in a deferred manner. (Ex. 1007, 6.) To limit communication overhead, clients learn about changes to the address table only when a client request reaches a wrong server, in which case the server forwards the client's request to the correct server and provides updates to the client's address table. (*Id.*, 5-6; Ex. 1002, ¶81.) Thus, Vingralek discloses updates to a client's stale address table are deferred until a wrong first server is reached and the location of a second server needs to be calculated.

Vingralek's method of deferred address table updates operates as follows. A hash function is applied to a key, which is used as an index to an address table listing available servers. (Ex. 1007, 9.) Each client and server has its own copy of the address table, but, for example, the client's table may have stale information if changes have been made to the servers since the table was last updated. (*Id.*, 10; *see also id.*, 11-13.) When a client seeks to retrieve records associated with a given key, it applies the hash function to the key, identifies the server where that

key is located by using the hashed key as an index into the current copy of the address table, and then sends the key in a request to that first server. (*Id.*, 9-10.) The first server that receives the key also applies the hash function to the key and identifies in its copy of the address table the server where the key resides (at least according to that first server's address table). (*Id.*, 10-11.) If the key was indeed sent to the correct server (i.e., if the identified server is the same as that first server), then that first server performs the requested operation. (*Id.*, 11.) But if the key was sent to the wrong server, the first server forwards the request to another server (i.e., the one identified by the above-described hash function application), and sends a message to the client to update the client's address table. (*Id.*; *see also id.*, 25 ("client requests and server responses" are "messages"); Ex. 1002, ¶82.)

The key is eventually received by the correct server and the request from the client is executed. (Ex. 1007, 11.) Via the address table update, "clients gradually bring their perception of the file up-to-date, so that the probability of wrongly addressed messages is significantly reduced." (*Id.*) Vingralek explains that if a server undergoes changes (e.g., if information is moved to another server), the server receives a new address table. (*Id.*, 12-13.) Consistent with the above, a POSITA would have understood that a server would have up-to-date information at least with respect to its own keys/information, such that the process of updating the client's address table each time a request is forwarded to the next server (until the

request reaches a “correct” server containing the requested information, i.e., “a location server known to have location information relevant to the location query”) would have gradually brought the client’s perception of the address table up-to-date. Thus, Vingralek discloses a first server calculating the location of a “correct” server to send a request to, and further discloses that if a first server does not have the requested information, a message (“redirect message”) is sent to the client with information to update the client’s address table. (Ex. 1002, ¶83.)

Additionally, Vingralek discloses that the “most salient feature” of its approach, where a client’s hash table is updated in a deferred manner, “is that its communication overhead is largely independent of the number of servers and clients in the system; thus, it is considered a scalable approach.” (Ex. 1007, 5.) Thus, Vingralek discloses implementing a redirection mechanism and additionally includes improved features to minimize the number of servers and reduce the cost of scaling up a distributed system. (*Id.*, 6, 8; Ex. 1002, ¶84.)

A POSITA would have been motivated to include features similar to those disclosed by Vingralek when implementing Skagerwall by providing a deferred address/hash table update to the client to limit communication overhead associated with updating hash tables in a scalable system. (Ex. 1002, ¶85.) Indeed, Skagerwall itself explains that, while an access server’s hash table is not always up to date, it is preferably not updated every time there is a change to the hash table,

so as to conserve resources. (Ex. 1006, 10:12-20.) As discussed above, Vingralek's method of updating the client's address table only when a request reaches the wrong server (i.e., in a deferred manner) is beneficial because it limits communication overhead, and improves the scalability of the network because communication overhead is independent of number of servers and clients in the system. (Ex. 1007, 5.) Thus, a POSITA would have found it beneficial to include Vingralek's method of deferred hash table updates, via a redirect message, when implementing Skagerwall's system/process. (Ex. 1002, ¶85.)

A POSITA would have further understood that it would also be beneficial for Vingralek's method of updating the client's address table via redirect message would also be beneficial to provide the addresses of "correct" location servers to an access server. (Ex. 1002, ¶86.) Skagerwall discloses that "[i]ntermediate steps are possible, where the look up request is relayed to further devices before it reaches the correct directory server." (Ex. 1006, 9:39-41.) Indeed, Skagerwall discloses an arrangement in which the first directory server ("queried location server") calculates the location of a correct directory server and the access server forwards the request to the correct directory server, but may involve an extra set of servers (referred to as broker servers), and does not explicitly disclose a specific process for transmitting the address of the correct directory server to the access server. (*Id.*, 16:47-51, 18:33-43.) Consistent with Vingralek's disclosure of updating the

client's address table, a POSITA would have been motivated to implement a system similar to that of Skagerwall where it includes in the redirect message only the portion of the address table corresponding to the location server(s) that contain the requested location information ("redirect message comprises information for finding a location server having location information related to the desired identifier"). This would allow the access server to update the relevant portions of its own hash/address table, consistent with Vingralek's teachings regarding the benefits of a deferred address table update, as well as provide the access server with the addresses of the location server(s) that should receive the forwarded request. (Ex. 1002, ¶86.)

A POSITA would have also understood that implementing Vingralek's redirect message feature would have reduced communication bandwidth in several ways, consistent with Vingralek's teachings of limiting communication overhead: by directly conveying the address of a correct directory server to the access server (without involving of an extra set of broker servers), and by including in the redirect message only the portion of the address table corresponding to the location server(s) that contain the requested location information, instead of the entire mapping/table. (Ex. 1002, ¶87.) Such a modification would have been beneficial as it would have reduced the frequency of communications to a client and/or the use of the client's computational resources, particularly in situations where the

client has limited communication bandwidth and/or computational resources (e.g., resources for calculating the correct directory server). (Ex. 1007, 5, 7, 8; Ex. 1006, 5:28-37, 10:15-20, 17:1-5, 18:54-58.)

A POSITA would have understood that this implementation, where a first directory server calculates a correct directory server and the access server forwards the request to that server (e.g. as disclosed by Skagerwall), would have been one of only several options, i.e., calculating and forwarding at either the access server or the directory server that processes the information request. (Ex. 1002, ¶88.) Thus, calculating the addresses of correct directory server(s) at a directory server and transmitting a redirect message, including those addresses, to the access server for the access server to forward the request to that correct directory server, would have been obvious because it would have been one of a “finite number of identified, predictable solutions.” *Perfect Web Techs., Inc. v. InfoUSA, Inc.*, 587 F.3d 1324, 1331 (Fed. Cir. 2009). As such, a POSITA would have been motivated to include features similar to Vingralek’s “redirect message” features when implementing Skagerwall.

A POSITA would have had reasonable expectation of success combining the teachings of Skagerwall and Vingralek. (Ex. 1002, ¶89.) As discussed above, both references are in the same field of art. Furthermore, Skagerwall discloses an “error message” that is provided if none of the directory servers stores the requested

object data (Ex. 1006, 13:5-6, 13:62-65), akin to the claimed “redirect message” that is provided if the location server does not contain the requested information.

In fact, a POSITA would have appreciated that the concept of providing a message when requested information is not found was well known. (*Id.*, 2:40-43.)

Moreover, Skagerwall itself suggests that its system/process is flexible and additional processes may be included, particularly with respect to the process of relaying a lookup request to a correct directory server. (*Id.*, 9:39-41.)

Additionally, a POSITA would have understood that implementing the combination would have involved modifying the Skagerwall system in a manner that Skagerwall suggests is feasible—implementing a means of occasionally updating a hash table to reduce communication overhead, and implementing a means of providing addresses of correct directory servers to the access server. (*Id.*, 10:12-20.) Moreover, such a modification would have involved applications of known technologies (e.g., known system of managing data and associated location information (Skagerwall)) according to known methods (e.g., known methods of occasionally updating hash tables using a redirect message (Vingralek)) to yield the predictable result of a system that updates hash tables in a manner that limits communication overhead and allows for scalability, as discussed above. *KSR*, 550 U.S. at 416. Accordingly, Skagerwall in view of Vingralek discloses and/or suggests this limitation. (Ex. 1002, ¶90.)

Consistent with the above and in such a combined system, for example, the access server would apply the hash function to the object tag and its own copy of the hash table, and send a request to the primary directory server (“queried location server”), which would apply the hash function to the object tag and its own copy of the hash table to determine whether the server has the requested location information. (Ex. 1002, ¶91.) If the primary directory server does not have the requested location information, e.g., application server addresses and application identifiers associated with the object tag (“if the queried location server does not contain location information relevant to the desired identifier”), the directory server would calculate the location/address of another directory server that has the requested location information (“information for finding a location server having location information related to the desired identifier”). (*Id.*) The directory server would then send a portion of the hash table corresponding to that directory server in a redirect message (“programming logic...responsive to a location query for a desired identifier to return...a redirect message...wherein the redirect message comprises information for finding a location server having location information related to the desired identifier”). (*Id.*) Such a redirect message (similar to Vingralek’s disclosed message to update a client’s hash table) would allow the access server to update its stale hash table, and additionally provide the access

server with the address of a correct directory server, which the access server uses to find the correct directory server and forward the request to that server. (*Id.*)

2. Claim 12

Skagerwall discloses this limitation. (Ex. 1002, ¶92.) As discussed for limitation [10.pre], the directory servers store object tags, application identifiers, and application server addresses (collectively “location information”), and each application server address indicates the location/address of an application server in a network (“location information comprises network address information”). (§IX.A.1.i.)

3. Claim 13

Skagerwall discloses this limitation. As discussed for limitation [10.pre] and claim 12, an application server address (“network address information”) indicates the network address of an application server (“data repository”), which contains application data packets (“data”) pertaining to a specific object tag (“desired identifier”). (§§IX.A.1.i, IX.A.2.) A POSITA would have understood that each application server is a “data repository” because it stores the application data packets (“data”) associated with tagged objects. (Ex. 1002, ¶93.)

4. Claim 14

i) [14.pre]

Skagerwall discloses this limitation. (Ex. 1002, ¶¶94-95.) **First**, Skagerwall discloses “a method of handling location queries in a network, the network comprising a plurality of location servers.” As discussed for limitation [10.pre], Skagerwall’s system includes a network of directory servers (“network comprising a plurality of location servers”), which store mappings of object tags, application server addresses, and application identifiers (collectively “location information”), which identify the location of application data packets. (§IX.A.1.i.) As further discussed, the system performs a method of handling requests for application server addresses and application identifiers (“location queries”) pertaining to a specific object tag, where the request for that location information is sent to a directory server (“location queries in a network”). (*Id.*)

Second, Skagerwall discloses “each location server containing a unique set of location information of an aggregate set of location information correlating each of a plurality of identifiers with at least one location.” Each directory server (“location server”) stores “a unique set of location information of an aggregate set of location information” for the reasons discussed for limitation [10.a]. (§IX.A.1.ii.) As discussed for limitation [10.pre], the location information stored on each directory server is stored in identifier/location mappings, such that each of

the object tags stored on a directory server (“each of a plurality of identifiers”) is associated with the corresponding application server addresses and application identifiers (together “at least one location”) for that object tag (“correlating each of a plurality of identifiers with at least one location”). (§IX.A.1.i; Ex. 1002, ¶95.)

ii) [14.a]

Skagerwall discloses this limitation. (Ex. 1002, ¶¶96-97.) As discussed for limitation [10.pre], the access server (“client”) transmits the information request (“location query”) to a directory server (“receiving a location query from a client at one of the plurality of location servers”). (§IX.A.1.i.) As further discussed, the request (“location query”) is for the application server addresses and application identifiers associated with the object tag that identifies an object (“entity”) in the request, where those application server addresses and application identifiers together specify the location of the application data packets (i.e., data) of an object (“requesting an entity’s location”). (*Id.*)

Skagerwall discloses the claimed “client” under the AWS Proceeding’s construction. (§VIII; Ex. 1001, 4:30-34.) A POSITA would have recognized that the access server (“client”) is a hardware and/or software component attached to a network given that Skagerwall explains that all components in the system may be “connected via a network N for communication with each other.” (Ex. 1006, 16:34-36; Ex. 1002, ¶97.) As discussed for limitation [10.pre], the access server

sends a request (i.e., location request message) to a directory server (i.e., location server) to initiate lookup of mappings of object tags (i.e., identifiers) to application identifiers and application server addresses (together providing location information about application data packets) from the directory server. (§IX.A.1.i.)

iii) [14.b]

Skagerwall discloses this limitation. (Ex. 1002, ¶98.) As discussed for limitation [10.pre], Skagerwall discloses a request (“query”) for the return of a mapping of an object tag to application server addresses and application identifiers (collectively “location information”) associated with an object (“entity”), where the request is sent to a primary directory server (“queried location server”) and includes the object tag that identifies an object (“entity identified in the query”). (§IX.A.1.i.) As discussed for limitation [10.b], if the directory server has the requested object tag, application server addresses, and application identifiers (“if the queried location server contains location information relevant to an entity identified in the query”), the directory server returns a message to the access server (“sending a location response message to the client”) with the object tag, associated application server addresses, and associated application identifiers (“the location response message comprising location information”). (§IX.A.1.iii.) As further discussed for limitations [10.pre] and [10.b], this sent location information identifies the location of application data packets of an object, with the application

server address identifying an application server that contains application data packets (i.e., data/information) associated with the object (“location information identifying at least one application server containing information relevant to the entity identified in the query”). (§§IX.A.1.i, IX.A.1.iii.)

iv) [14.c]

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶99.) As discussed for limitation [10.pre], Skagerwall discloses a request (“query”) for object tags, application server addresses, and application identifiers (collectively “data location information”)⁶ associated with an object (“entity”), where the request is sent from the access server to a primary directory server (“queried location server”) and includes the object tag that identifies an object (“entity identified in the query”). (§IX.A.1.i.) As discussed for limitation [10.b], the Skagerwall-Vingralek combination discloses and/or suggests that if the queried directory server (“queried location server”) does not have the object tags, application server addresses, and application identifiers (collectively “data location information”) that were requested (“if the queried location server

⁶ Skagerwall discloses “data location information” under the AWS Proceeding’s construction of “location information” for the same reasons as discussed for limitation [10.pre]. (§§VIII, IX.A.1.ii.)

does not contain data location information relevant to the entity identified in the query”), the directory server returns a message to the access server (“sending a redirect message to the client”) that identifies which of the other directory servers stores the requested application server addresses and application identifiers associated with the object tag (“the redirect message comprising information for finding a location server storing the entity identified in the query”). (§IX.A.1.iii.)

5. Claim 15

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶100.) As discussed for limitation [10.b], in the Skagerwall-Vingralek combination, a queried directory server (“queried location server”) applies the object tag identified in the request (“location query”) from the access server (“applying information in the location query from the client”) to a hash function (“indexing function”) to determine whether the directory server contains the object tag, application server addresses, and application identifiers mapping (collectively “location information”) associated with the object tag (identifying an object (“entity”)) included in the request (“to determine if the queried location server contains location information relevant to the entity identified in the query”). (§IX.A.1.iii.) A POSITA would have understood that an indexing function is a function that returns the index to a table, and thus the hash function of the

Skagerwall-Vingralek combination, which returns a value used as an index to the hash table, is an “indexing function.” (Ex. 1002, ¶100.)

6. Claim 16

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. Skagerwall discloses that each directory server (“each of the plurality of location servers”) has a memory means that stores object tags (“identifiers”) and the associated application server addresses and application identifiers (both “associated locations”). (Ex. 1006, 15:15-20, 15:49-51 (“all object data...stored in the provided memory means.”).) A POSITA would have understood that each directory server’s memory means comprises an “indexed table of identifiers and associated locations” because each object tag (“identifier”) is used as an index to retrieve all associated application server addresses and application identifiers (both “associated locations”). (Ex. 1002, ¶101.) As discussed for claim 15, Skagerwall also discloses that “applying information in the location query” comprises applying an object tag (“identifier”) in the request (“location query”) to a hash function. (§IX.A.5.)

7. Claim 17

i) [17.pre]

To the extent the preamble of claim 17 is limiting, Skagerwall in view of Vingralek discloses and/or suggests the limitations therein. (Ex. 1002, ¶¶102-109.)

Skagerwall discloses “a system having a plurality of location servers for storing and retrieving location information, wherein each of the plurality of location servers stores [a] unique set of location information of an aggregate set of location information.” As discussed for limitation [10.pre], Skagerwall discloses a system including directory servers (“a system having a plurality of location servers”) for storing and retrieving application server addresses and application identifiers associated with an object tag (collectively “location information”). (§IX.A.1.i.) Skagerwall also discloses that each directory server (“each of the plurality of location servers”) stores “a unique set of location information of an aggregate set of location information” for the reasons discussed for limitation [10.a]. (§IX.A.1.ii; Ex. 1002, ¶102.)

Skagerwall does not explicitly disclose “a method of scaling at least one of capacity and transaction rate capability in a location server” in the system. A POSITA would have been motivated and found it obvious to implement this feature in view of Vingralek. (Ex. 1002, ¶103.)

As discussed for limitation [10.b], Vingralek and Skagerwall are in the same field of art, and a POSITA would have been motivated to consider the teachings of Vingralek when implementing Skagerwall’s system/method. (§IX.A.1.iii; Ex. 1002, ¶104.) As discussed therein, Vingralek discloses a system where a file is organized into buckets spread across servers. (§IX.A.1.iii; Ex. 1007, 4.) Vingralek

employs a local load control at each server, preventing server overloading by redistributing buckets when a specified server utilization is reached. (Ex. 1007, 6.) Vingralek also implements an “explicit mapping of buckets onto servers,” allowing “additional flexibility to redistribute the load of an overloaded server” by, e.g., splitting bucket(s) of the server, migrating a bucket to a newly-acquired server, or migrating a bucket to an existing server. (*Id.*, 6-7.) Vingralek determines the utilization/performance capacity of the existing servers and decide on splitting versus migrating buckets, and acquiring new servers versus redistributing across existing servers. (*Id.*, 7.) Vingralek explains that a logically centralized server called a file advisor monitors/controls utilization/performance capacity. (*Id.*, 7, 12.)

Vingralek describes two load capacity concepts: the “feasible load capacity,” i.e., “the maximum number of keys that the server can keep without being overloaded,” and the “panic load capacity,” i.e., “the number of keys after which a server is no longer capable of accepting additional keys.” (*Id.*, 10.) Vingralek describes several load management techniques for splitting/migrating buckets, and for moving buckets to an existing server or acquiring a new server. (*Id.*, 11-14.) For example, if a server exceeds its feasible load capacity, the file advisor begins monitoring that server’s capacity (*id.*, 12) until the server reaches the panic load capacity, upon which the file advisor identifies a second server having enough

unused performance capacity to accept some buckets from the first at-capacity server, and sends a migrate message to the first server, triggering the transfer of buckets from the first server to another (second) server. (*Id.*, 13.) Thus, Vingralek discloses scaling the capacity of a server by, e.g., increasing the effective capacity by transferring buckets of information to another server when certain thresholds (e.g., load capacity concepts) are reached. (Ex. 1002, ¶105.)

Additionally, Vingralek discloses that its method of scaling “strives for the objective of ‘cost-conscious’ scalability, by aiming to minimize the number of servers...used for providing the necessary performance capacity” (Ex. 1007, 6) and improves on previous systems because they does not consider “the cost of scaling up a distributed system..., and thus no attempt is made to minimize the number of servers across which a file is spread” (*id.*, 8). (Ex. 1002, ¶106.) In particular, Vingralek discloses that its approach is scalable because new servers are acquired (e.g., as a way of managing the server load) only if server load metrics exceed certain thresholds. (Ex. 1007, 24.)

A POSITA would have been motivated to include scalability features similar to those disclosed by Vingralek when implementing a system similar to that of Skagerwall to manage the directory servers’ load capacities while minimizing the number of servers storing location data, so as to reduce the cost of scaling up the distributed system. (Ex. 1002, ¶107.) For example, when a directory server is no

longer able to accept additional requests to add object tags (e.g., reaching panic load capacity as discussed in Vingralek), consistent with Vingralek’s teaching, a POSITA would have found it obvious to move/transfer certain object tag(s) and associated object data, including application server addresses and application identifiers (both “associated locations”), to a second/another directory server to allow the system of directory servers to continue fulfilling requests. Additionally, a POSITA would have found it beneficial to implement Vingralek’s teaching as it strives for the objective of cost-conscious scalability by not acquiring new servers unless the load on existing servers exceeds certain thresholds. (Ex. 1007, 10-13.)

Moreover, a POSITA would have found Vingralek’s method beneficial because while Skagerwall discloses the database of directory servers can be scaled arbitrarily (Ex. 1006, 4:12-15, 8:15-19), it does not explicitly disclose the specific processes for scaling that system of servers, e.g., how or when a server’s load capacity is exceeded, which Vingralek teaches. A POSITA would have been motivated to include features similar to Vingralek’s method of scaling server capacity when implementing Skagerwall’s distributed system of directory servers, such that the combined system includes a method of scaling the capacity of a directory server in the distributed system of directory servers (“[a] method of scaling at least one of capacity and transaction rate capability in a location server in a system having a plurality of location servers”). (Ex. 1002, ¶108.)

A POSITA would have had reasonable expectation of success combining the teachings of Skagerwall and Vingralek. As discussed above, both references are in the same field of art. (*Supra.*) Moreover, Skagerwall discloses that its directory servers may be scaled arbitrarily. (Ex. 1006, 4:12-15, 8:15-19) Thus, combining the teachings of Skagerwall and Vingralek would have involved modifying the Skagerwall system in a manner that Skagerwall suggests is feasible—implementing a means of scaling the distributed database of directory servers. Additionally, the combination of Skagerwall and Vingralek would have involved applications of known technologies (e.g., known distributed systems (Skagerwall)) according to known methods (e.g., known methods of scaling and managing load capacity of a distributed system (Vingralek)) to yield the predictable result of a system implementing a method of scaling the distributed database of directory servers, as discussed above. *KSR*, 550 U.S. at 416. Accordingly, Skagerwall in view of Vingralek discloses and/or suggests this limitation. (Ex. 1002, ¶109.)

ii) [17.a]

iii) [17.b]

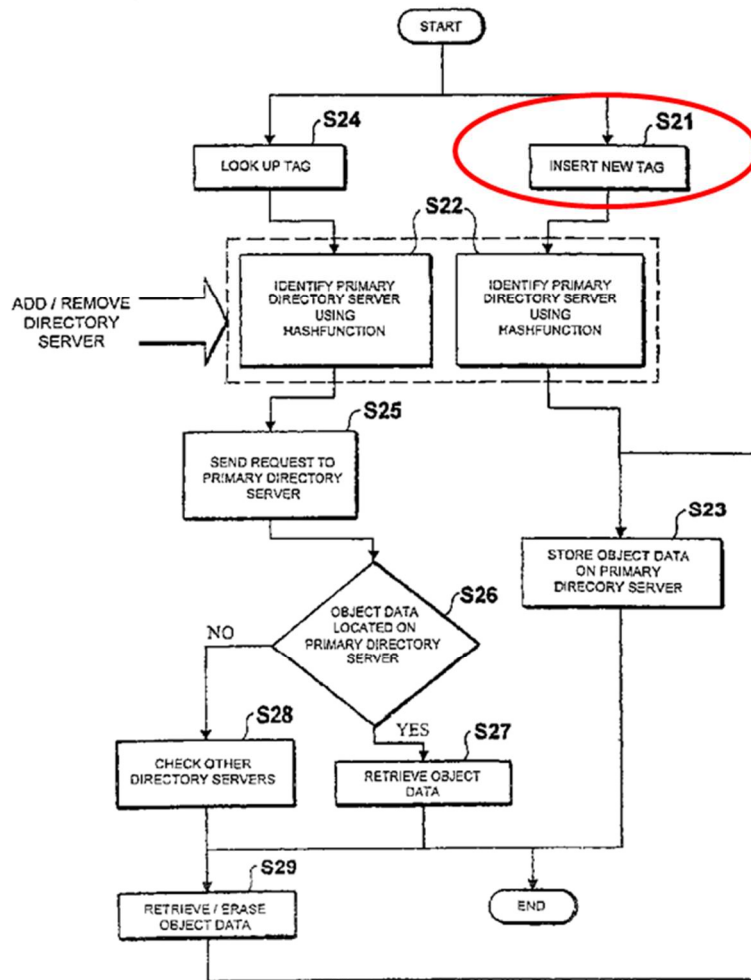
iv) [17.c]

Skagerwall discloses and/or suggests these limitations. (Ex. 1002, ¶¶110-116.) **First**, Skagerwall discloses and/or suggests “transport[ing] identifier and location information, the location information specifying the location of

information related to the identifier,” “storing location information...at a first location server,” and “receiving an identifier and a location relevant to the identifier at the first location server.” As discussed for limitation [10.pre], Skagerwall discloses a network of directory servers (each a “location server”), where each server stores mappings of object tags (“identifier”), application server addresses, and application identifiers (the three types of information in the mapping collectively “location information”), where such a mapping specifies the location of application data packets (“information”) associated with an object tag (“location information specifying the location of information related to the identifier”). (§IX.A.1.i; Ex. 1002, ¶110.)

With respect to Figure 2a, Skagerwall describes inserting new object data into the database of object data distributed across directory servers. (Ex. 1006, 12:3-4.) Skagerwall discloses that a request for insertion of new object data (including a mapping of object tag, application server addresses, and application identifiers (collectively “location information”)), corresponding to an object tag (“identifier”) that is included in the request, is received by the system kernel in step S21 (red) (“transport identifier and location information, the location information specifying the location of information related to the identifier”). (*Id.*, 12:5-7, 14:29-56.)

FIG.2a



(*Id.*, FIG. 2A.) In step S22, a target directory server (“first location server”) for storing the object data is identified by applying a hash function to the object tag (“identifier”) and using the result as an index for searching the hash table. (*Id.*, 12:8-21, 14:37-53.) In step S23, the request to store the object data (including the disclosed “location information”) associated with a specific object tag (“identifier”) is sent to the target directory server (“transport identifier and location information, the location information specifying the location of information related

to the identifier”; “receiving an identifier and a location relevant to the identifier at the first location server”), which performs steps for storing/inserting the object data (“storing location information...at a first location server”). (*Id.*, 12:22-26; Ex. 1002, ¶111.)

To the extent Skagerwall does not explicitly disclose transporting both “identifier” and “location information” to a directory server as part of the request to insert new object data, a POSITA would have found it obvious to implement Skagerwall’s system in this manner. As discussed for limitation [10.pre], the directory servers each store object tags, application server addresses, and application identifiers in the form of a set of mappings, which map object tags (“identifier”) to all associated object data (including all the application server addresses and application identifiers associated with a specific object tag).

(§IX.A.1.i.) Thus, a POSITA would have found it obvious to modify Skagerwall’s system to implement the request to add new object data such that the request includes the object tag itself and any associated object data to be added. (Ex. 1002, ¶112.) A POSITA would have found it beneficial to implement this feature because aligning the insertion and storage of object data would have allowed for simplified storage of the object tag/object data mappings, without requiring additional steps (beyond the insertion request) to associate the object tag and object data on the directory server. In such a modified system, when a directory server

receives a request to add new application server addresses and application identifiers (both “location[s]”), that request would include the object tag (“identifier”) and the application server addresses and application identifiers themselves to update the set of mappings (“transport identifier and location information, the location information specifying the location of information related to the identifier”; “receiving an identifier and a location relevant to the identifier at the first location server”). (*Id.*)

A POSITA would have also had reasonable expectation of success in implementing this modification in the Skagerwall-Vingralek combination. (Ex. 1002, ¶113.) Implementing features similar to those disclosed in Skagerwall would have involved modifying the Skagerwall-Vingralek combination in a manner that Skagerwall suggests is feasible—aligning the processes for insertion and storage of new object data. Indeed, Skagerwall discloses storing the object data on the directory servers as part of the insertion process (Ex. 1006, 12:1-26), and also discloses storing the object data alongside the associated object tag (*id.*, 8:43-63, 15:15-20, 15:49-52). Moreover, such a combination would have involved applications of known technologies (e.g., known systems for managing data and associated location information (Skagerwall)) according to known methods (e.g., known methods of inserting and storing location information (Skagerwall)) to yield the predictable result of a communication system in which object data insertion

requests include both the object tag and associated object data. *KSR*, 550 U.S. at 416.

Second, Skagerwall discloses and/or suggests “**providing a transfer protocol configured to transport identifier and location information**, the location information specifying the location of information related to the identifier.” Because Skagerwall discloses the access server constitutes the link between the user and system kernel (Ex. 1006, 16:39-40), a POSITA would have understood that the above-described request to insert object location information would have been sent from the access server to a directory server. (Ex. 1002, ¶114.) Thus, for reasons similar to as discussed for limitation [10.a], a POSITA would have formatted the insertion request, like the information request discussed for limitation [10.a], according to known protocols, such as HTTP (“providing a transfer protocol configured to transport identifier and location information”), given that Skagerwall discloses that its system may be used with the Internet and requests may be made via HTTP. (§IX.A.1.ii; Ex. 1006, 9:59-62, 18:2-7, 18:24-26.)

Third, Skagerwall discloses “**storing location information formatted according to the transfer protocol** at a first location server.” As discussed above and for limitation [10.a], Skagerwall discloses and/or suggests formatting communications between access server and directory server (e.g., communications

including object data) according to a transfer protocol such as HTTP. (§IX.A.1.ii.)

To the extent Skagerwall does not disclose *storing* object data according to a transfer protocol, a POSITA would have found it obvious to store the disclosed “location information,” at least temporarily, according to the format the information is transmitted in, i.e., HTTP standards, at the directory server (“storing location information formatted according to the transfer protocol at a first location server”). Moreover, as discussed for limitation [10.a], it would have been obvious to retrieve the “location information” in the HTTP format/standards. (§IX.A.1.ii.) Thus, a POSITA would have also found it beneficial to store such information according to the format that it is transmitted in. (Ex. 1002, ¶115.)

A POSITA would have also had reasonable expectation of success in implementing this modification in the Skagerwall-Vingralek combination. (*Id.*, ¶116.) Implementing features similar to those disclosed in Skagerwall would have involved modifying the Skagerwall-Vingralek combination in a manner that Skagerwall suggests is feasible—implementing the system in line with a transfer protocol. (Ex. 1002, ¶116; Ex. 1006, 9:59-62, 18:2-7, 18:24-26.) Moreover, such a combination would have involved applications of known technologies (e.g., known communication systems (Skagerwall)) according to known methods (e.g., known transfer protocols (Skagerwall)) to yield the predictable result of a

communication system in which object data is stored at a directory server according to known transfer protocols (i.e., HTTP). *KSR*, 550 U.S. at 416.

v) [17.d]

Skagerwall discloses and/or suggests this limitation. (Ex. 1002, ¶117.) As discussed for claim 16, Skagerwall discloses that each directory server (“data location server”)⁷ has a memory means (“location store”) that stores object tags (“comprising a plurality of identifiers”) and the application server addresses and application identifiers (both “location[s]”) associated with each object tag (“each identifier associated with at least one location”). (§IX.A.6.) A POSITA would have thus understood that when a target directory server (“first data location server”) stores the received object tag (“the received identifier”) and associated application server addresses and application identifiers (both “the received location”), the directory server would store the information in the memory means (“location store”) such that the object tag is associated with the corresponding location information (“storing the received location in a location store at the first

⁷ Skagerwall discloses a “data location server” under the AWS Proceeding’s construction of a “location server” for the same reasons as discussed for limitation [10.pre]. (§§VIII, IX.A.1.ii.)

data location server...wherein the received location is associated with the received identifier in the location store”). (Ex. 1002, ¶117.)

vi) [17.e]

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶118.) As discussed for limitation [17.pre], in the Skagerwall-Vingralek combination, if a directory server (“first data location server”) reaches the panic load capacity (“when a performance criterion of the first location server reaches a predetermined performance limit”), it would have been obvious to move/transfer certain object tag(s) and associated object data to a second/another directory server to allow the system of directory servers to continue fulfilling requests. (§IX.A.7.i.) As discussed for limitation [10.pre], the object data includes associated application server addresses and application identifiers (both “locations”). (§IX.A.1.i.) Thus, the Skagerwall-Vingralek combination discloses and/or suggests transferring a portion of the object tags (“identifiers”) and associated application server addresses and application identifiers (both “locations”) to the second directory server (“transferring a portion of the identifiers and associated locations to a second data location server”) when the panic load capacity is reached (“when a performance criterion of the first location server reaches a predetermined performance limit”). (Ex. 1002, ¶118.)

8. Claim 23

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. As discussed for limitations [17.pre] and [17.e], the panic load capacity (“performance criterion”) is “the number of keys after which a server is no longer capable of accepting additional keys.” (Ex. 1007, 10, 21 (“[T]he panic load capacity of a server is expressed in terms of the maximum number of keys after which the server would start thrashing.”); §§IX.A.7.i and IX.A.7.vi.) While Vingralek expresses this capacity as a number of keys (similar to Skagerwall’s object tags, i.e., identifiers) that a server may hold, Vingralek explains that it assumes that the key size is a fixed/constant amount of storage space. (Ex. 1007, 21 (“[T]he record size and the key size were fixed throughout the simulation.”), 22 (displaying simulation parameter settings in which the panic load capacity is 11000 records, record size is 10 Kbytes, and key size is 100 Bytes).) Thus, a POSITA would have recognized that in the Skagerwall-Vingralek combination the panic load capacity factors in and/or represents the amount of persistent storage space available on a directory server (“the performance criterion comprises an amount of available persistent storage space in the first location server”). (Ex. 1002, ¶119.)

9. Claim 30

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. As discussed for limitation [17.pre], Vingralek discloses that after a server reaches the feasible load capacity, the file advisor begins monitoring that server to determine if it hits other thresholds, e.g., the panic load capacity (“monitoring the performance criterion”). (§IX.A.7.i.) When a server hits the panic load capacity (“when the first location server reaches the predetermined limit”), the file advisor begins identifying a second server for transferring buckets/information to, and sending a migrate message to the first server to trigger transferring of the information (“automatically transferring” information “when the first location server reaches the predetermined limit”). (*Id.*) For reasons similar to those discussed for limitation [17.pre], a POSITA would have found it obvious to implement in the Skagerwall-Vingralek combination processes similar to those described above, to monitor the directory server’s load capacity and automatically transfer the object tags (“the portion of identifiers”) and associated object data (e.g., application server addresses and application identifiers (both “associated locations”)) when the monitored server reaches the panic load capacity (“when the first location server reaches the predetermined limit”). (Ex. 1002, ¶120.)

**B. Ground 2: Skagerwall in View of Vingralek and Krasner
Renders Claim 11 Obvious**

1. Claim 11

Skagerwall in view of Vingralek and Krasner discloses and/or suggests this limitation. (Ex. 1002, ¶¶121-125.) Skagerwall discloses the directory servers may more broadly store object data “associating each of the objects with at least one of the application data packets.” (Ex. 1006, 4:13-32.) While the Skagerwall-Vingralek combination does not expressly disclose that the directory servers may store geographic location information, a POSITA would have found it obvious to implement this feature in view of Krasner. (Ex. 1002, ¶121.)

Like Skagerwall, Krasner relates to a client-server system. Krasner discloses a system including location servers that store location information pertaining to a mobile GPS unit. (Ex. 1009, Abstract, 4:10-15.) Thus, Krasner is similar to Skagerwall that stores location information pertaining to objects. (Ex. 1006, 7:37-43, 8:15-28.) A POSITA would have therefore been motivated to consider Krasner when implementing Skagerwall’s system/method. (Ex. 1002, ¶122.)

Krasner additionally discloses a “system for tracking the location of a mobile GPS receiver over time.” (Ex. 1009, 3:56-57; *see also id.*, 1:48-51, 2:39-41; Ex. 1002, ¶123.) The mobile GPS unit receives GPS signals at various positions and determines pseudoranges from those signals, which are stored with

corresponding time stamps. (Ex. 1009, 5:49-53.) Krasner's system includes a location server communicating with the mobile GPS unit. (*Id.*, 4:10-13.) The mobile GPS unit transmits the pseudoranges and time stamps to the location server when certain event(s) occur, e.g., after certain time period or emergency. (*Id.*, 4:66-5:24, 5:49-6:15, 6:54-7:23.) The location server uses the transmitted information to "provide concierge services or routing information" to a user. (*Id.*, 7:23-28.) Thus, Krasner discloses a location server storing geographical location information ("the location information comprises geographic location information") of mobile GPS unit(s) and providing such information to a user.

A POSITA would have been motivated to include such features when implementing Skagerwall's system. Skagerwall discloses a flexible system where a directory server ("location server") stores object data, including various location information (e.g., object tags, application server addresses, application identifiers) that provide the location of application data packets (i.e., data) associated with an object. (Ex. 1006, 8:15-28.) When object data associated with a specific object tag is retrieved from a directory server, the object data may be directly transmitted to a user. (*Id.*, 4:20-32, 7:49-52, 8:28-37, 12:54-62.) Thus, a POSITA would have been motivated to consider modifying Skagerwall's system to store different types of object data on directory servers, or modifying to enable certain applications in which object data (e.g., location information) is shared directly with a user. As

discussed above, Krasner discloses location servers (akin to Skagerwall's directory servers) storing geographical location information and providing that information directly to users. A POSITA would have recognized such a feature would have beneficially allowed a user to track an object's location over time. (Ex. 1002, ¶124.) Indeed, Skagerwall's objects may "be any products or items of the physical world." (Ex. 1006, 6:45-48, 9:49-50.) It would have been beneficial to track the geographical locations of physical objects, e.g., products purchased or vehicles, and allow a user to request the tracked geographic location information via a server, e.g., a directory server, using an object tag identifying the object (i.e., mobile GPS unit), similar to as described for limitation [10.pre]. (§IX.A.1.i.)

A POSITA would have also had reasonable expectation of success combining the Skagerwall-Vingralek combination with Krasner as implementing the above-described features would have involved applications of known technologies (e.g., known system (Skagerwall)) according to known methods (e.g., known methods of storing geographic location information and providing the same (Krasner)) to yield the predictable result of a system storing geographic location information on directory servers and providing such information per request. (Ex. 1002, ¶125.) *KSR*, 550 U.S. at 416.

C. Ground 3: Skagerwall in View of Vingralek and Sato Renders Claims 24-26 Obvious

1. Claim 24

Skagerwall in view of Vingralek and Sato discloses and/or suggests this limitation. (Ex. 1002, ¶¶126-130.) As discussed for limitations [17.pre] and [17.e], the Skagerwall-Vingralek combination discloses and/or suggests scaling capacity of a location server, and transferring certain identifiers and associated locations to a second data location server when a performance criterion of the first location server reaches a predetermined performance limit. (§§IX.A.7.i, IX.A.7.vi.) While the Skagerwall-Vingralek combination does not disclose scaling a location server’s capability where the performance criterion comprises a transaction rate limit, a POSITA would have found it obvious to implement this feature in view of Sato. (Ex. 1002, ¶126.)

Like Skagerwall and Vingralek, Sato describes a “client-server system built on a distributed system” where the client issues requests to server(s). (Ex. 1011, ¶[0003].) Sato uses another set of servers, i.e., service location servers, which provide location information of the specific server that provide services to clients. (*Id.*) Sato discloses a server partitioning method that partitions the service range and manages location information in a distributed manner “when the load on the service location server exceeds a predetermined threshold.” (*Id.*, ¶[0005].) Thus, Sato is similar to Vingralek, which discloses migrating information from one

server to another when the first server reaches a performance criterion threshold.

(Ex. 1007, 5-8, 13.) A POSITA would have therefore been motivated to consider Sato when implementing the Skagerwall-Vingralek combination. (Ex. 1002, ¶127.)

Sato additionally discloses that each “service location server” includes a “load measuring means” measuring server’s “service load information” (Ex. 1011, ¶[0009]), where the measured information includes “the maximum number of service executions per unit time...after starting the service of the server” (*id.*, ¶[0012]). Whether to partition the server “is determined by comparing the maximum number of service executions [per unit time]...with [a predetermined] threshold value” (*id.*, ¶¶[0013], [0029]), and if partitioning is necessary, then information on the server is migrated to another server (*id.*, ¶¶[0029]-[0030]). Thus, Sato discloses scaling transaction rate capability in a server, where information is transferred from a first to second server when the maximum number of service executions per unit time (or execution rate) reaches a predetermined threshold value (“performance criterion comprises a transaction rate limit”). (Ex. 1002, ¶128.)

A POSITA would have been motivated to include features similar to those disclosed by Sato when implementing the Skagerwall-Vingralek combination. (Ex. 1002, ¶129.) Vingralek assumes the “server instruction rate” (i.e., server

processing speed) is constant. (Ex. 1007, 22.) While Vingralek generally describes a homogenous system (*id.*, 21), its approach is “applicable also to heterogeneous servers...hav[ing] different load capacities” (*id.*, 26). Thus, a POSITA would have been motivated to modify the Skagerwall-Vingralek system to account for heterogeneous servers where the server instruction rate (processing speed) varies based on the characteristics of each server. Sato’s method of partitioning a server based on a maximum service execution rate is advantageous because it allows for dynamic distributions of server workload upon reaching a predetermined threshold, rather than requiring distribution in advance, thereby “increasing the flexibility of the system.” (Ex. 1011, ¶[0073].) Sato’s method would have been beneficial for allowing implementation of a partitioning method in a system of heterogeneous servers, where processing speed may vary.

Consistent with the above and in such a combined system, each directory server would be configured to have a set maximum service execution rate threshold such that, in addition to transferring identifier/location mappings to a second directory server based on the load capacity principles discussed in Vingralek (e.g., panic load capacity), it would also transfer identifier/location mappings to a second directory server when a first server’s maximum service execution rate exceeds a predetermined threshold. (Ex. 1002, ¶129.)

A POSITA would have also had reasonable expectation of success combining the teachings of Skagerwall, Vingralek, and Sato. (Ex. 1002, ¶130.) As discussed above, these references are in the same field of art. (*See supra.*) Moreover, it would have involved modifying the Skagerwall-Vingralek system in a manner that Vingralek suggests is feasible—implementing methods of partitioning servers in a heterogeneous system. Additionally, the combination of Skagerwall, Vingralek, and Sato would have involved applications of known technologies (e.g., known system for partitioning servers (Vingralek)) according to known methods (e.g., known methods of partitioning when a maximum service execution rate exceeds a threshold (Sato)) to yield the predictable result of a system in which directory servers are configured to transfer information to other servers when a transaction rate limit is reached, as discussed above. *KSR*, 550 U.S. at 416.

2. Claim 25

The Skagerwall-Vingralek-Sato combination discloses and/or suggests this limitation. As discussed for claim 24, the performance criterion is a maximum service *execution* rate (“transaction rate limit”). (§IX.C.1.) A POSITA would have understood that the maximum service execution rate reflects the maximum speed at which a processor of a directory server operates (“transaction rate limit comprises a processor speed limit”). (Ex. 1006, 15:1-7; Ex. 1002, ¶131.)

3. Claim 26

The Skagerwall-Vingralek-Sato combination discloses and/or suggests this limitation. (Ex. 1002, ¶¶132-134.) Vingralek discloses the **performance** of the distributed network is “characterized by the network latency *NL* (i.e., time required to process each packet...) and the **bandwidth** *BW* (i.e., transmission time needed to send one byte through the network)” (Ex. 1007, 22), which a POSITA would have understood is of a *finite* value/number. Thus, Vingralek discloses “a network connection bandwidth limit.” A POSITA would have understood the disclosed “network connection bandwidth limit” is a factor of the “transaction rate capability” (limitation [17.pre]) or “transaction rate limit” (claim 26) of a directory server (“location server”) because a network’s bandwidth impacts the rate at which the servers receive requests from clients, and thus server(s)’ maximum service execution rate. (Ex. 1002, ¶132; §IX.A.7.i.)

With the above, a POSITA, when implementing the Skagerwall-Vingralek-Sato combination, would have found it obvious to “transfer[] a portion of the identifiers and associated locations to a second data location server when a performance criterion of the first location server reaches a predetermined performance limit” (limitation [17.e]), where the “performance criterion” is “the transaction rate limit compris[ing] a network connection bandwidth limit” (claim 26). (§IX.A.7.i.) The bandwidth of the part of the network connected to a first

directory server may be limited/reduced due to, e.g., available bandwidth being occupied by an increased number of client requests. This is particularly true as the servers may be heterogeneous having different load capacities as disclosed in Vingralek. (Ex. 1007, 26; §IX.C.1.) Thus, a POSITA would have found it beneficial to conduct a process similar to as described for limitation [17.e], e.g., identifying another directory server having sufficient network connection bandwidth to accept buckets from a first directory server and transferring the same. (§IX.A.7.vi.) It would have reduced the load of the first directory server, and increased the second directory server's utilization, consistent Vingralek's "objective of 'cost-conscious' scalability, by aiming to minimize the number of servers...used for providing the necessary performance capacity." (Ex. 1007, 6; Ex. 1002, ¶133.)

A POSITA would have also had reasonable expectation of success in implementing such feature in the Skagerwall-Vingralek-Sato combination. (Ex. 1002, ¶134.) It would have involved modifying the combination in a manner Vingralek suggests is feasible—implementing methods of partitioning servers in a heterogeneous system. The modification would have involved applications of known technologies (e.g., known system for partitioning servers and known network bandwidth limit (Vingralek)) according to known methods (e.g., known methods of partitioning when a maximum service execution rate exceeds a

threshold (Sato)) to yield the predictable result of a system where directory servers are configured to transfer information to other servers when reaching a network connection bandwidth limit. *KSR*, 550 U.S. at 416.

X. DISCRETIONARY DENIAL IS NOT APPROPRIATE

A. The Board Should Not Deny Institution Under § 325

Discretionary denial under § 325(d) is not appropriate here because this petition does not present the same or substantially the same prior art or arguments previously presented to the Office during prosecution. (Ex. 1004; Ex. 1001, Cover.) Moreover, there are material differences between the art/arguments here and those during prosecution (*e.g.*, Ex. 1004, 305-314), including features believed to be missing (*id.*, 334-342). Discretionary denial is also not warranted in view of the completed *ex parte* reexamination proceedings and IPR, since this petition does not present the same or substantially the same prior art or arguments presented in those proceedings, and there are material differences between the art/arguments here with those presented in those proceedings (*e.g.*, Ex. 1042, 23-24; Ex. 1043, 12-14; Ex. 1047, 15-18), including features believed to be missing (Ex. 1022, 19-23; Ex. 1045, 9-10; Ex. 1049, 8-10.) And the Office erred in a material manner by allowing the claims despite such features being known. *Advanced Bionics*, Paper 6 at 10-11; *Becton, Dickinson*, Paper 8.

Discretionary denial is also not warranted in view of the pending *ex parte* reexamination initiated by a different party, AWS. (Ex. 1051; §II.) First, Petitioner had no involvement with the reexamination, and Petitioner developed the present petition independently of AWS.

Second, this petition and the reexamination are based on materially different grounds and positions. None of the positions in the reexamination relies on Vingralek, which is asserted in each of these grounds. (Ex. 1051, 17-43; §§IV, IX.) Thus, there is no substantial overlap between the grounds here and those remaining in the pending EPR.

Third, the pending claims of the reexamination and the challenged claims in this IPR are not coextensive. (§§II, IV; Ex. 1051.)

Fourth, denying institution in view of a third-party reexamination would be unfairly prejudicial. The reexamination does not challenge all of the claims that are challenged in this proceeding. Moreover, it is Patent Owner that decided to accuse separate companies of infringement, each with separate interests in challenging the patent. And, given the statutory deadline to file an IPR, Google could not have waited until the conclusion of the reexamination. *See Mueller Sys., LLC v. Rein Tech, Inc.*, IPR2020-00100, Paper 9 at 12-14 (May 12, 2020) (“Petitioner would be barred from challenging claims...following the Reexamination Proceeding.”).

Fifth, the reexamination remains at an early stage, which counsels against discretionary denial. *Mueller* at 13.

B. Discretionary Denial Under *General Plastic* Is Not Appropriate

Petitioner has never before filed a Patent Office challenge to the '978 patent. Patent Owner may argue that the '978 patent is already at issue in proceedings initiated by different parties. (§II.) The facts here, however, do not support a discretionary denial.

The Board has set forth seven factors in determining whether to exercise discretion to deny a petition due to an earlier proceeding. *Gen. Plastic Indus. Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 15-19 (Sept. 6, 2017) (precedential) (“*General Plastic*”).

With respect to the first factor, Petitioner is not a party, real party-in-interest, or privy to the other Patent Office proceedings and this is Petitioner’s first challenge to the '978 patent, which “weighs especially heavily against a discretionary denial.” *Unified Patents, Inc. v. Certified Measurement, LLC*, IPR2018-00548, Paper 7 at 7-8 (PTAB Sept. 5, 2018). Moreover, unlike *Valve*, AWS and Petitioner were sued at different times—four and a half years apart—and the AWS-challenged claims (§II) and the challenged claims in this IPR are not coextensive. *Valve Corp. v. Elec. Scripting Prods., Inc.*, IPR2019-00062, Paper 11 at 10 (PTAB April 2, 2019) (precedential).

The second to fifth factors “bear little relevance” here because Petitioner has never before challenged the ’978 patent. *Unified* at 7-8. Petitioner has no relationship to the earlier challenges to the ’978 patent. And there is no indication that AWS was aware of the prior art combination at issue here—the Skagerwall-Vingralek combination. In fact, the complaint of the district court litigation against Petitioner was filed in June 2023, several months after AWS filed its currently pending reexamination request in February 2023. (Ex. 1050, 228.)

As to the sixth factor, while the Board has finite resources, instituting this petition would be no more a burden on these finite resources than instituting any other petition. And for the seventh factor, there is no readily identifiable roadblock for the Board to issue a final determination within the statutory one-year limit.

C. The Board Should Not Deny Institution Under *Fintiv*

The merits of Petitioner’s arguments are compelling and the evidence in support is substantial. That “alone demonstrates that the PTAB should not discretionarily deny institution under *Fintiv*.” (Ex. 1036, 4-5.) In any event, the six *Fintiv* factors do not justify denying institution.

The **first factor** (stay) weighs strongly against denial. Although Google has not yet moved to stay in district court pending IPR, the Northern District of Illinois has stayed a number of cases pending IPR proceedings since 2020. (Ex. 1033, 1, 2, 5, 6, 10.)

The **second factor** (trial proximity) weighs strongly against denial, since “the median time-to-trial is...after the projected...final written decision.” (Ex. 1036 (Memorandum by Director Vidal), 9.) Specifically, the median time-to-trial in the Northern District of Illinois is close to 57 months. (Ex. 1035 (Federal Court Management Statistics), 47.) The district court case was filed on June 30, 2023, so trial based on the statistics would occur in April 2028. On the other hand, with this petition filed in June 2024, a final written decision may be expected around December 2025. Moreover, this petition was diligently filed shortly after Patent Owner served its infringement contentions identifying asserted claims on April 25, 2024. (Ex. 1034, 1.)

The **third factor** (investment in parallel proceedings) also weighs strongly against denial. The district court case is in the early stages. (*Id.*, 1.) The date of claim construction hearing is still to be determined but is likely to be no earlier than April/May 2025. (*Id.*, 2.) Moreover, the district court’s discovery process is currently stayed pending resolution of Google’s Motion to Dismiss until July 15, 2024. (Ex. 1052, 1; Ex. 1055, 1.)

The **fourth factor** (overlap) also weighs against denial, since Petitioner challenges more claims than Patent Owner may ultimately assert in its final contentions in the district court litigation, i.e., 10 claims per patent and 20 claims

overall from the '640, '170, and '978 patents. (Ex. 1053, 10.) *Apple, Inc. v. SEVEN Networks, LLC*, IPR2020-00156, Paper 10 at 17, 21-22 (June 15, 2020).

The **sixth** factor (other circumstances) also weighs against denial as the merits of Petitioner's arguments are compelling. While the **fifth** factor (parties) may weigh slightly in favor of denial because the parties are the same as in district court, that is true in almost every IPR, and under the "holistic view" of whether integrity of the system and efficiency is best served, institution is favored.

Samsung Elecs. Co. Ltd. v. Dynamics Inc., IPR2020-00505, Paper 11 at 15 (Aug. 12, 2020).

XI. CONCLUSION

Petitioner requests institution of IPR for the challenged claims based on the grounds specified in this petition.

Respectfully submitted,

Dated: June 28, 2024

By: /Naveen Modi/
Naveen Modi (Reg. No. 46,224)
Counsel for Petitioner

XII. APPENDIX A (CLAIM LISTING)

Claim 10

[10.pre] “A system having a plurality of location servers for managing location information and providing location information to location queries, the system comprising:”

[10.a] “a plurality of location servers containing location information corresponding to a plurality of entities, the location information formatted according to a transfer protocol configured for manipulating location information, and comprising at least one application server address, wherein the plurality of location servers are arranged in a cluster topology such that each location server contains a unique set of location information of an aggregate set of the location information; and”

[10.b] “programming logic stored on each of the plurality of location servers responsive to a location query for a desired identifier to return one of a location message, wherein a queried location server returns a location message if the queried location server contains location information for the desired identifier, and a redirect message if the queried location server does not contain location information relevant

to the desired identifier, wherein the redirect message comprises information for finding a location server having location information related to the desired identifier.”

Claim 11

“The system of claim 10, wherein the location information comprises geographic location information.”

Claim 12

“The system of claim 10, wherein the location information comprises network address information.”

Claim 13

“The system of claim 12, wherein the network address information comprises a network address of a data repository containing data pertaining to the desired identifier.”

Claim 14

[14.pre] “A method of handling location queries in a network, the network comprising a plurality of location servers, each location server containing a unique set of location information of an aggregate set of location information correlating each of a plurality of identifiers with at least one location, the method comprising:”

[14.a] “receiving a location query from a client at one of the plurality of location servers, the location query requesting an entity’s location;”

[14.b] “sending a location response message to the client if the queried location server contains location information relevant to an entity identified in the query, the location response message comprising location information identifying at least one application server containing information relevant to the entity identified in the query; and”

[14.c] “sending a redirect message to the client if the queried location server does not contain data location information relevant to the entity identified in the query, the redirect message comprising information for finding a location server storing the entity identified in the query.”

Claim 15

“The method of claim 14, further comprising applying information in the location query from the client to an indexing function to determine if the queried location server contains location information relevant to the entity identified in the query.”

Claim 16

“The method of claim 15, wherein each of the plurality of location servers comprises an indexed table of identifiers and associated locations, and wherein applying information in the location query comprises applying an identifier in the location query to a hash function.”

Claim 17

[17.pre] “A method of scaling at least one of capacity and transaction rate capability in a location server in a system having a plurality of location servers for storing and retrieving location information, wherein each of the plurality of location servers stores unique set of location information of an aggregate set of location information, the method comprising:”

[17.a] “providing a transfer protocol configured to transport identifier and location information, the location information specifying the location of information related to the identifier;”

[17.b] “storing location information formatted according to the transfer protocol at a first location server;”

[17.c] “receiving an identifier and a location relevant to the identifier at the first location server;”

[17.d] “storing the received location in a location store at the first data location server, the location store comprising a plurality of identifiers, each identifier associated with at least one location, wherein the received location is associated with the received identifier in the location store; and”

[17.e] “transferring a portion of the identifiers and associated locations to a second data location server when a performance criterion of the first location server reaches a predetermined performance limit.”

Claim 23

“The method of claim 17, wherein the performance criterion comprises an amount of available persistent storage space in the first location server.”

Claim 24

“The method of claim 17, wherein the performance criterion comprises a transaction rate limit.”

Claim 25

“The method of claim 17, wherein the transaction rate limit comprises a processor speed limit.”

Claim 26

“The method of claim 17, wherein the transaction rate limit comprises a network connection bandwidth limit.”

Claim 30

“The method of claim 17, wherein transferring a portion of the identifiers and associated locations to a second location server when a performance criterion of the first data location server reaches a predetermined performance limit further comprises monitoring the performance criterion and automatically transferring the portion of identifiers and associated locations when the first location server reaches the predetermined limit.”

CERTIFICATE OF COMPLIANCE

Pursuant to 37 C.F.R. § 42.24(d), the undersigned certifies that the foregoing
Petition for *Inter Partes* Review of U.S. Patent No. 7,233,978 contains, as
measured by the word-processing system used to prepare this paper, 13,968 words.
This word count does not include the items excluded by 37 C.F.R. § 42.24 as not
counting towards the word limit.

Respectfully submitted,

Dated: June 28, 2024

By: /Naveen Modi/
Naveen Modi (Reg. No. 46,224)
Counsel for Petitioner

CERTIFICATE OF SERVICE

I hereby certify that on June 28, 2024, I caused a true and correct copy of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 7,233,978 and supporting exhibits to be served via express mail on the Patent Owner at the following correspondence address of record as listed on Patent Center:

Reichman Jorgensen Lehman & Feldberg LLP
100 Marine Parkway
Suite 300
Redwood City, CA 94065

A courtesy copy was also sent via electronic mail to the Patent Owner's litigation counsel at the following addresses:

Christine E Lehman (clehman@reichmanjorgensen.com)
Jaime Francisco Cardenas-Navia (jcardenas-navia@reichmanjorgensen.com)
Gina H Cremon (gcremona@reichmanjorgensen.com)
Khue Hoang (khoang@reichmanjorgensen.com)

Reichman Jorgensen Lehman & Feldberg LLP

Respectfully submitted,

By: /Naveen Modi/
Naveen Modi (Reg. No. 46,224)
Counsel for Petitioner