

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner

v.

KOVE IO, INC.,
Patent Owner

Patent No. 7,814,170

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,814,170**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	MANDATORY NOTICES	1
III.	PAYMENT OF FEES UNDER 37 C.F.R. §§ 42.15 AND 42.103	3
IV.	GROUND FOR STANDING.....	3
V.	PRECISE RELIEF REQUESTED AND GROUNDS RAISED.....	3
VI.	LEVEL OF ORDINARY SKILL	4
VII.	OVERVIEW OF THE '170 PATENT	5
VIII.	CLAIM CONSTRUCTION	8
IX.	DETAILED EXPLANATION OF GROUNDS.....	12
A.	Ground 1: Kahn in View of Vingralek Renders Claims 1-17 Obvious.....	12
1.	Claim 1	12
2.	Claim 2	31
3.	Claim 3	32
4.	Claim 4.....	35
5.	Claim 5	36
6.	Claim 6	38
7.	Claim 7	52
8.	Claim 8	54
9.	Claim 9	54
10.	Claim 10	54
11.	Claim 11	55
12.	Claim 12	56
13.	Claim 13	57
14.	Claim 14	58
15.	Claim 15	59
16.	Claim 16	62

17. Claim 17.....	63
X. DISCRETIONARY DENIAL IS NOT APPROPRIATE	64
A. The Board Should Not Deny Institution Under § 325	64
B. Discretionary Denial Under <i>General Plastic</i> Is Not Appropriate.....	66
C. The Board Should Not Deny Institution Under <i>Fintiv</i>	68
XI. CONCLUSION.....	69
XII. APPENDIX A (CLAIM LISTING)	71
Claim 1	71
Claim 2	72
Claim 3	72
Claim 4	72
Claim 5:	72
Claim 6	73
Claim 7	74
Claim 8	74
Claim 9	74
Claim 10	75
Claim 11	75
Claim 12	75
Claim 13	75
Claim 14	76
Claim 15	76
Claim 16	77
Claim 17	77

LIST OF EXHIBITS

Ex. 1001	U.S. Patent No. 7,814,170 to Overton et al. (“the ’170 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. 7,814,170
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)
Exs. 1008-1010	RESERVED
Ex. 1011	Japanese Patent Application Publication No. H09-231180 to Sato et al.
Exs. 1012-1013	RESERVED
Ex. 1014	RFC 768, User Datagram Protocol (Aug. 28, 1980)
Ex. 1015	M. C. Norrie et al., <i>The Database Research Group at ETH Zurich</i> , SIGMOD Record (March 1994)
Ex. 1016	U.S. Patent No. 5,764,756
Exs. 1017-1018	RESERVED
Ex. 1019	First Amended Complaint (Dkt. No. 36) in <i>Kove IO, Inc. v. Google LLC</i> , Case No. 1:23-cv-04244 (N.D. Ill. Nov. 14, 2023)
Ex. 1020	Declaration of Sylvia D. Hall-Ellis, Ph.D.
Ex. 1021	RESERVED

Petition for *Inter Partes* Review
Patent No. 7,814,170

Ex. 1022	<i>Amazon Web Services, Inc. v. Kove IO, Inc.</i> , IPR2020-00276, Paper 23 (June 16, 2020)
Ex. 1023	RESERVED
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)
Ex. 1037	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,814,170 in <i>Ex Parte</i> Reexamination No. 90/019,035 (Nov. 19, 2021)
Ex. 1038	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,035 (Jan. 12, 2022)
Ex. 1039	Notice of Intent to Issue <i>Ex Parte</i> Reexamination Certificate in <i>Ex Parte</i> Reexamination No. 90/019,035 (Sept. 27, 2022)
Ex. 1040	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,814,170 in <i>Ex Parte</i> Reexamination No. 90/019,165 (Feb. 17, 2023)
Ex. 1041	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,165 (Mar. 29, 2023)

Exs. 1042-1051	RESERVED
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	RESERVED
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 1-17 (“the challenged claims”) of U.S. Patent No. 7,814,170 (“the ’170 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 ’170 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 ’170 patent was involved in ex-parte reexamination No. 90/019,035 (“the ’035 EPR”), and is currently pending in ex-parte reexamination No. 90/019,165 (“the ’165 EPR”).

The ’035 EPR was filed by third party Amazon Web Services (“AWS”) (Ex. 1037) in November 2021, presenting SNQs on claims 1, 2, 6, 8, 9, 12, and 15 based on different combinations of prior art from the grounds here (*id.*, 11-14;

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.

infra Sections IV, IX); the request was granted (Ex. 1038, 21-22) and the claims were confirmed (Ex. 1039, 16). Petitioner had no involvement in the preparation or filing of the '035 EPR.

The '165 EPR was filed by third party AWS (Ex. 1040) in February 2023, presenting SNQs on claims 1, 2, 6, 8, 9, 12, and 15 based on different combinations of prior art from the grounds here (*id.*, 19-21; *infra* Sections IV, IX.). Petitioner had no involvement in the preparation or filing of the '165 EPR. Because the request raised substantial new questions of patentability, reexamination was ordered in March 2023. (Ex. 1041, 19.) 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 7,233,978 (“the '978 patent”), which are also at issue in the above-noted district court cases. The '978 patent was also at issue in *Amazon Web Services, Inc. v. Kove IO, Inc.*, IPR2020-00276 (“the AWS IPR”).

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) that the '170 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 1-17 of the '170 patent and cancellation of these claims as unpatentable on the following grounds:

Ground 1: Claims 1-17 are unpatentable under pre-AIA 35 U.S.C. § 103 as being obvious over Kahn (Ex. 1005) and Vingralek (Ex. 1007).

The earliest possible effective filing date for the '170 patent is July 8, 1998.²

² Petitioner believes the '170 patent is not entitled to a priority date earlier than September 14, 1999, but does not raise that issue in this petition as it is not relevant. Petitioner does not concede the '170 patent is entitled to either the July 8, 1998 or September 14, 1999 priority date.

Kahn published on November 20, 1997 from PCT application U.S. 1997/007834 filed on May 9, 1997. (Ex. 1005, Cover). Thus, Kahn qualifies as prior art at least under pre-AIA 35 U.S.C. § 102(a).

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.)

VI. LEVEL OF ORDINARY SKILL

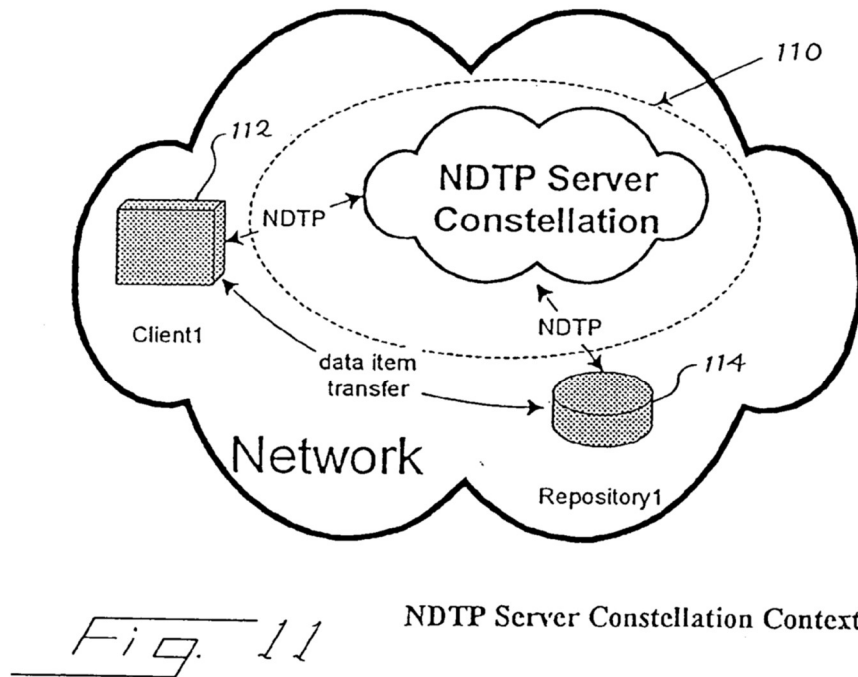
A person of ordinary skill in the art at the time of the alleged invention of the '170 patent (“POSITA”) would have had an undergraduate degree in electrical engineering, computer engineering, computer science or a related field along with two years of work experience in the field of distributed computing systems. (Ex.

1002, ¶¶20-21.)³ More education can supplement practical experience and vice versa. (*Id.*)

VII. OVERVIEW OF THE '170 PATENT

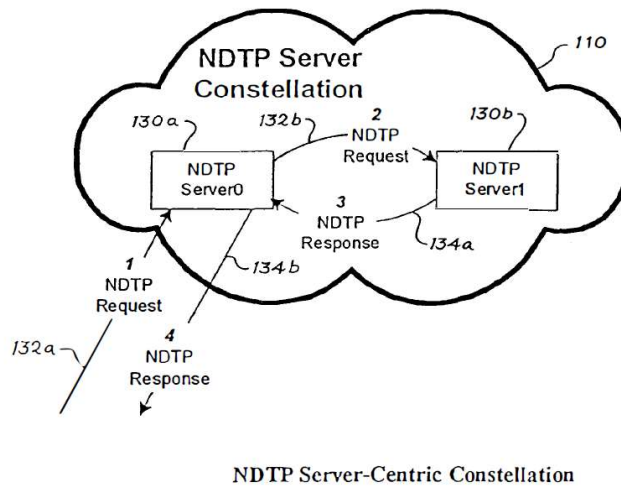
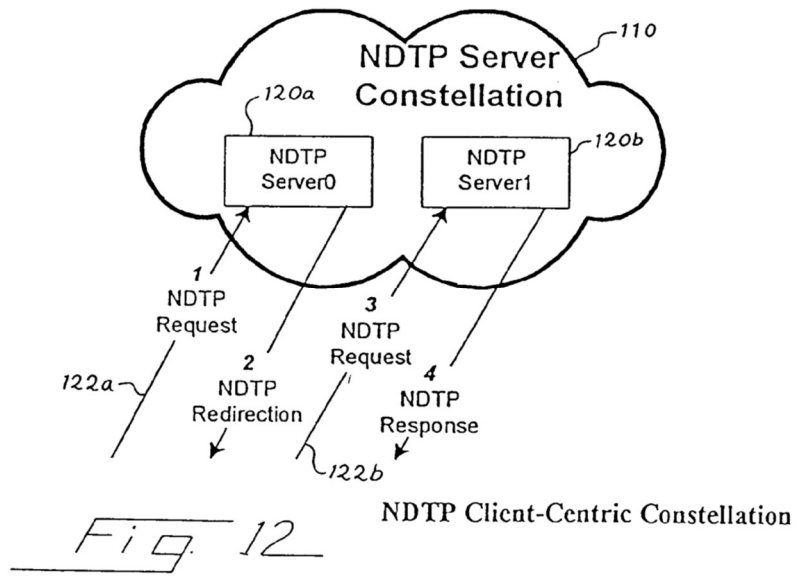
The '170 patent describes storing and retrieving information across a network (Ex. 1001, 1:29-32, 1:58-60) and a network distributed tracking protocol (NDTP) (*id.*, 2:14-16, 4:28-38). Figure 11 shows an NDTP server constellation where a client consults servers that return strings in response to a client request. (*Id.*, 17:3-6.)

³ Petitioner submits the declaration of Dr. Sandeep Chatterjee (Ex. 1002), an expert in the field of the '170 patent. (Ex. 1002, ¶¶1-28; Ex. 1003.)



(*Id.*, FIG. 11; Ex. 1002, ¶29.)

Server constellations have two organizational approaches: client-centric (Figure 12) and server-centric (Figure 13). In the client-centric approach, processing emphasis is toward clients (*id.*, 18:25-27), whereas in the server-centric approach, server processing is emphasized (*id.*, 18:50-54).



(*Id.*, FIGS. 12, 13; Ex. 1002, ¶30.)

In both approaches, a client transmits an identifier specifying an entity to a server in a client request. (Ex. 1001, 2:14-45.) 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 '170 patent were known in the prior art before the alleged invention date of the '170 patent. (*Infra* Section IX; Ex. 1002, ¶¶22-28, 32-38; Exs. 1006, 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, ¶¶39-47.)⁴

As shown below, in the AWS IPR, the Board construed a claim term of the related '978 patent (Ex. 1022, 13), and in the AWS Proceeding, the district court

⁴ Petitioner reserves all rights to raise claim construction and other arguments in other venues.

construed several claim terms of the '170 patent. (Ex. 1024, 35; Ex. 1025, 19.)

While Petitioner does not acquiesce to those constructions, the prior art mappings in Section IX explain how the '170 patent claims are disclosed and/or rendered obvious if those constructions are applied. (Ex. 1002, ¶¶39-47.)

'170 Patent Terms	AWS Proceeding/IPR Construction
“identifier” ⁵	“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”	“information pertaining to one or more locations of data and/or the identities of one or more location servers” (Ex. 1024, 35.)

⁵ Petitioner applies the construction of “identifier” recited in the '978 patent claims to '170 patent claims reciting “identifier string” because the '170 patent uses the two terms interchangeably (Ex. 1001, 4:10-12) and describes the terms in substantially the same manner as the '978 patent. (*Compare* Ex. 1001, 4:10-12, *with* Ex. 1022, 10-11.)

'170 Patent Terms	AWS Proceeding/IPR Construction
“location server”	“a network-attached component that maintains a set of identifier/location mappings that are modified or returned 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.)
“the portion of the data location information included in a corresponding one of the data location servers is <i>based on a hash function used to organize the data location</i>	“the portion of the data location information included in a corresponding one of the data location servers is <i>based on a hash function that maps identifier strings to one or</i>

'170 Patent Terms	AWS Proceeding/IPR Construction
<i>information across the plurality of data location servers</i> , and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string” ⁶	<i>more of the data location servers</i> , and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string” (Ex. 1024, 35.)
“data pertaining to the entity”	“data pertaining to a person or thing (real, digital, or abstract) distinct from that data” (Ex. 1024, 35.)
“plurality of location servers” ⁷	“location servers in non-hierarchical structures” (Ex. 1025, 19.)

⁶ Petitioner applies only the bolded and italicized part of this construction, as the rest of the construction recites the exact claim language.

⁷ Petitioner also applies this construction to claims reciting “plurality of data location servers.”

IX. DETAILED EXPLANATION OF GROUNDS

As detailed below, each of the challenged claims are unpatentable. (Ex. 1002, ¶¶39-47, 48-128.)

A. Ground 1: Kahn in View of Vingralek Renders Claims 1-17 Obvious

1. Claim 1

i) [1.pre]

To the extent the preamble is limiting, Kahn discloses the limitations therein. (Ex. 1002, ¶¶49-51.) To begin, Kahn discloses a system for managing digital objects stored in one or more repositories in a network, where the digital objects are located by one or more pointers associated with a handle. (Ex. 1005, Abstract (“A method of managing digital objects...in a network is presented.”), 1:10-12 (“By a ‘digital object’ we broadly mean any set of sequences of bits or digits and an associated unique identifier which we call a ‘handle.’”), 6:9-10, 6:24.) Multiple handle servers in the network accept handles of digital objects and return corresponding pointers indicating the digital objects’ locations in the network. (Ex. 1005, Abstract, 6:28-35, 7:2-20, 21:6-24; Ex. 1002, ¶49.) Kahn’s Figure 1 shows an exemplary system including a repository (blue) for storing digital objects (green) in the network, where digital objects are identified by their handle (yellow) and associated pointers. (Ex. 1005, Abstract.) Handle-pointer mappings are stored

in tables within handle servers 1042 (red). (*Id.*, Abstract, 7:19-20, 11:16-18, 37:31-33.)

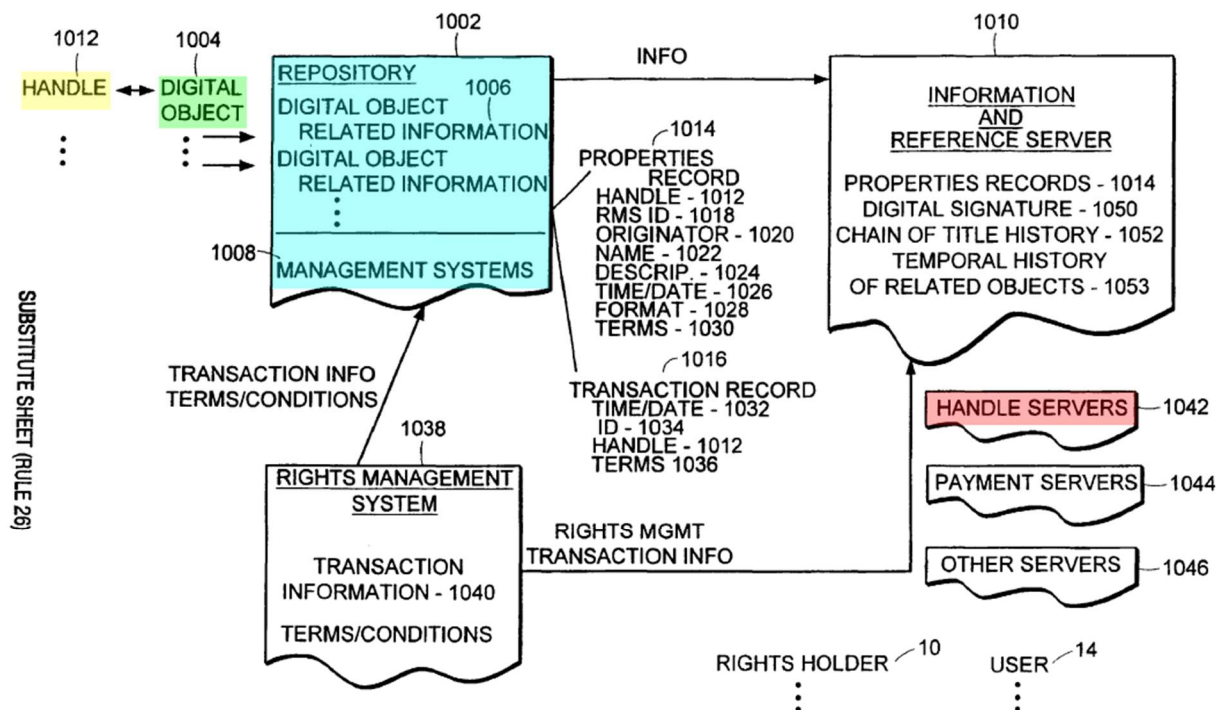


FIG. 1

(*Id.*, FIG. 1; Ex. 1002, ¶49.)

Kahn discloses that digital objects (including “data”) are “stored in a distributed network.” A digital object is represented by a sequence of bits/digits and may include conventional digital representations of real-world objects (e.g., books, images, sounds, software). (Ex. 1005, 1:10-12, 1:16-20.) A POSITA would have understood that sequences of bits/digits associated with each digital object are “data.” (Ex. 1002, ¶50.) Indeed, “a digital object could include programs and data” which enable delivery over a network or reproduction on a

computer screen. (Ex. 1005, 1:20-25.) Kahn discloses that a digital object is stored in a repository in a distributed network, such as the Internet, which does not have a central nodal point. (*Id.*, Abstract, 2:25-30, 6:9-35, 11:16-18, 21:21-24, 73:23-31; Ex. 1001, 1:58-67, 4:50-57.)

Moreover, Kahn discloses that its communication system is for “managing data stored in a distributed network.” For example, Kahn discloses managing digital objects (including “data”) in a distributed network. (Ex. 1005, Abstract, 2:25-30, 6:9-35, 11:16-18, 21:21-24, 73:23-31.) As further described for limitation [1.b], Figure 8 (below) shows a client system retrieving one or more pointers for digital objects. (*Id.*, FIG. 8, 46:26-28, 46:29-47:25, 72:32-74:32; Section IX.A.1.iii; Ex. 1002, ¶51.)

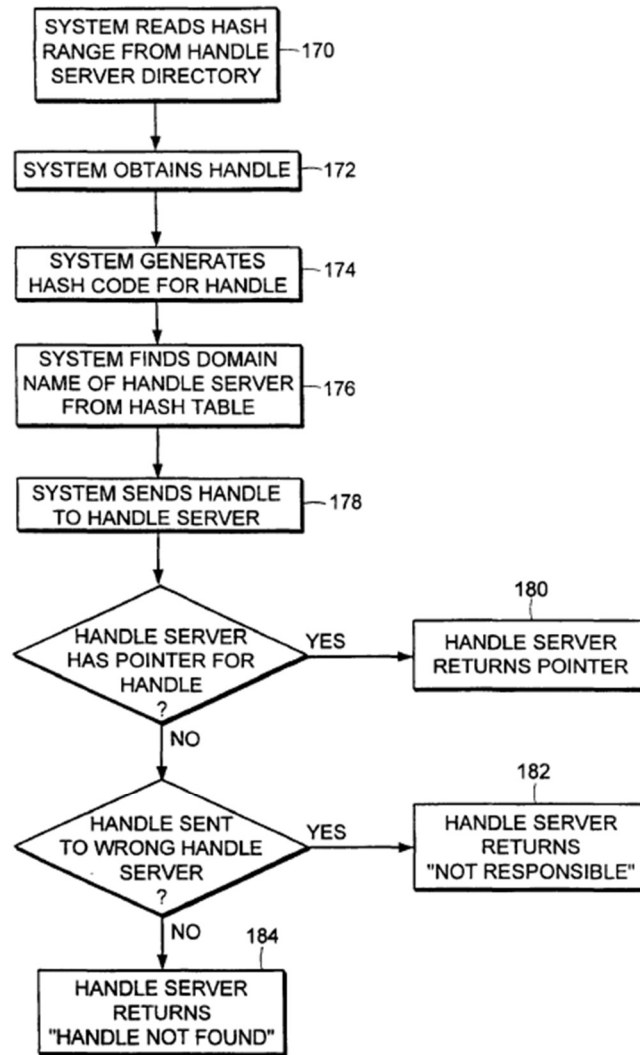


FIG. 8
SUBSTITUTE SHEET (RULE 26)

(Ex. 1005, FIG. 8; Ex. 1002, ¶51.)

ii) [1.a]

Kahn discloses this limitation. (Ex. 1002, ¶¶52-54.) **First**, Kahn discloses “a data repository configured to store a data entity.” As discussed for limitation [1.pre], digital objects (“data entity”) are stored in one or more individual repositories (together a “data repository”) in a network. (Ex. 1005, Abstract, 6:9-

7:2, 11:16-18, 21:21-24, 73:23-31, FIG. 1; *see* Section IX.A.1.i.) The sequences of bits/digits (“data”) pertaining to a particular object constitute the claimed “data entity.” (Section IX.A.1.i; Ex. 1002, ¶52.)

Second, Kahn discloses “wherein an identifier string identifies the data entity.” (Ex. 1002, ¶53.) Kahn discloses that each digital object (“data entity”) is identified by a unique handle (“identifier string”), where the “handles comprise unique character strings.” (Ex. 1005, 7:4-5, 21:3-5 (“Each digital object has a **‘handle’, a concise unique identifier for a digital object...**”); *see also id.*, 6:27-28, 9:23-26, 11:8-12.)

Kahn’s handle is an “identifier string” under the AWS IPR construction of related term “identifier.” (Section VIII.) A POSITA would have understood that a handle uniquely identifies a real-world object (i.e., an individual entity) because a handle uniquely identifies a digital object (i.e., a data entity), and as discussed for limitation [1.pre], the digital object is a digital representation of a real-world object/work. (Section IX.A.1.i; Ex. 1002, ¶54.) A POSITA would have further understood that the handle is a unique encoding that identifies the object because the object and its digital representation (i.e., digital object) are uniquely represented and encoded in the system by the characters that comprise the handle. (Ex. 1002, ¶54; Ex. 1005, 33:3-14, 33:22-27; Sections IX.A.1.i, IX.A.1.iii.) As further discussed for limitation [1.b], the handle is associated with zero or more

pointers (i.e., data location specifiers) in a handle server (i.e., a location server, as construed in the AWS Proceeding). (Section IX.A.1.iii.)

iii) [1.b]

Kahn in view of Vingralek discloses and/or suggests this limitation. (Ex. 1002, ¶¶55-72.) **First**, Kahn discloses “a data location server network comprising a plurality of data location servers.” As discussed for limitation [1.pre], Kahn discloses a network containing multiple handle servers (“a data location server network comprising a plurality of data location servers”). (Section IX.A.1.i; Ex. 1005, 7:5-9, 21:6-24; Ex. 1002, ¶55.)

Kahn also discloses the claimed “data location server[]” under the AWS Proceeding’s construction of “location server.” (Section VIII; Ex. 1001, 4:17-20.) Kahn discloses that each handle server (“location server”) is a component attached to a distributed network (i.e., a network-attached component). (Ex. 1005, 7:5-7, 21:21-24, 73:27-31.) Each handle server maintains mappings between handles (i.e., identifiers) and pointers (i.e., locations). (*See* Section IX.A.1.i; Ex. 1005, 7:17-20 (“...multiple handle servers, each...having a table which associates handles with pointers.”), 21:6-10, 37:31-33.)⁸ Further, Kahn discloses that the

⁸ Kahn’s pointer discloses the “location” (or similar terms) as described in the ’170 patent (Ex. 1001, 4:12-14) given that pointers are associated with certain handles

handle server returns identifier/location mappings in response to a request from the client for location information associated with a handle (i.e., in response to location request messages from a client). For example, as discussed further below, pointers (i.e., locations) are returned to the client if the handle server determines it contains the requested location information associated with the handle in the request. Given that the returned pointers are associated with the handle in the location information request, a POSITA would have understood the returned pointers correspond to mappings of the handle to the pointers, consistent with the '170 patent's disclosure. (Ex. 1002, ¶56; Ex. 1001, 2:38-42, 2:61-65, 6:67-7:4, 17:3-6.) Moreover, Kahn discloses that the full handle record (i.e., including the handle itself and all corresponding data) may be returned in response to a request from the client. (Ex. 1005, 42:1-11 (“client request may ask for all the data in a handle record...”), 44:16-21.) Thus, a POSITA would have understood that if pointers are returned to the client from the handle server, the handle itself may also be returned in response to the location request message from the client. (Ex. 1002, ¶56.) Kahn also discloses that identifier/location mappings may be updated/modified in response to a request from the client to add new location information (i.e., another type of

and both are stored in a handle server (Ex. 1005, 7:17-20, 21:6-10, 37:31-33; Ex. 1002, ¶56.)

location request message from the client). (Ex. 1005, 43:31-44:4, 45:22-46:24; Ex. 1001, 6:50-53.)

Further, Kahn discloses the claimed “plurality of data location servers” under the AWS Proceeding’s construction of “plurality of location servers.” (Section VIII.) A POSITA would have understood that the handle servers (“data location server[s]”) 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. (Ex. 1002, ¶57.) Kahn discloses that the hash function is used to distribute handles and associated pointers among the handle servers. (Ex. 1005, 4:19-22, 7:8-9, 37:31-38:3, FIG. 7.) A POSITA would have thus understood that because a hash function is used for distribution across handle servers, Kahn’s handle servers are arranged in a non-hierarchical configuration. (Ex. 1002, ¶57.) 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.)

Second, Kahn discloses “wherein data location information for a plurality of data entities is stored in the data location server network.” Kahn discloses that handles and their associated pointers (“data location information”), which indicate the locations of digital objects (each including “data,” and together a “plurality of data entities”), are stored in multiple handle servers in the network (“data location

server network”). (Ex. 1005, 28:31-34 (“A handle for an object may be associated with zero or more object pointers...contain[ing] location information for locating digital objects.”), 37:31-33; *see also id.*, Abstract, 3:21-22, 6:30-35, 28:32-34; Section IX.A.1.i.) For the same reason discussed above, Kahn discloses the claimed “data location information” and “location information” under the AWS Proceeding’s construction of “location information.” (Section VIII; Ex. 1002, ¶58.)

Third, Kahn discloses “at least one of the plurality of data location servers includes location information associated with the identifier string.” For example, Kahn discloses each handle server (“at least one of the plurality of data location servers”) has a table associating pointers (“location information”) with a handle (“identifier string”). (Ex. 1005, 7:17-20, 21:8-10, 37:31-33; Section IX.A.1.i; Ex. 1002, ¶59.)

Fourth, Kahn discloses “each one of the plurality of data location servers comprises a processor and a portion of the data location information.” Kahn discloses each handle server (“each one of the plurality of data location servers”) provides handle-to-pointer translation services, i.e., a service returning a pointer in response to an associated handle, where each handle server can be on one or more computers operating using software. (Ex. 1005, 21:6-15, 26:28-31, 29:5-12, 42:2-11, 44:6-7, 65:14-69:14.) Thus, a POSITA would have understood that each

handle server (“each one of the plurality of data location servers”), which may be a computer, includes a “processor.” Further, Kahn discloses each handle server stores a portion of all pointers (“portion of the data location information”) for locating digital objects. (Ex. 1005, 7:5-9, 37:31-38:3 (“a handle server holds pointers associated with a subset of all handles”), 46:22-24; Ex. 1002, ¶60.)

Fifth, Kahn discloses “the portion of the data location information included in a corresponding one of the data location servers is based on a hash function used to organize the data location information across the plurality of data location servers.” For example, Kahn discloses that the portion of pointers (“the portion of the data location information”) included in a corresponding handle server (“one of the data location servers”) is based on the application of a hash algorithm (“hash function”), as discussed further below with respect to Figure 7, used to partition/distribute the handles and associated pointers (“organize the data location information”) across the multiple handle servers (“across the plurality of data location servers”). (Ex. 1005, 37:31-38:3 (“[H]andles are assigned to handle servers based upon hash values computed on the handles....”), 47:1-11 (“The system then generates the hash code for the handle using a predetermined hashing algorithm....”); *see also id.*, 29:23-26; Ex. 1002, ¶61.)

Kahn discloses that the portion of data location information is included in a corresponding data location server “based on a hash function used to organize the

data location information across the plurality of data location servers” under the AWS Proceeding’s construction. (Section VIII.) Kahn discloses that its hash algorithm maps handles (“identifier string[s]”) to one or more handle servers (“data location servers”) because the hash values generated by applying the hash algorithm to the handle are used to partition/distribute the handles amongst multiple handle servers. (Ex. 1005, 29:23-26, 37:31-38:3, 47:1-11; Ex. 1002, ¶62.)

Sixth, Kahn discloses “determin[ing] the at least one of the plurality of data location servers based on the hash function applied to the identifier string.” Figure 8 (below) shows a process by which a client retrieves one or more pointers for digital objects, which involves applying a hash function to a handle (“hash function applied to the identifier string”) to determine which handle server to send a request to (“determine the at least one of the plurality of data location servers”). (Ex. 1005, FIG. 8, 46:26-28, 46:29-47:25, 72:32-74:32.)

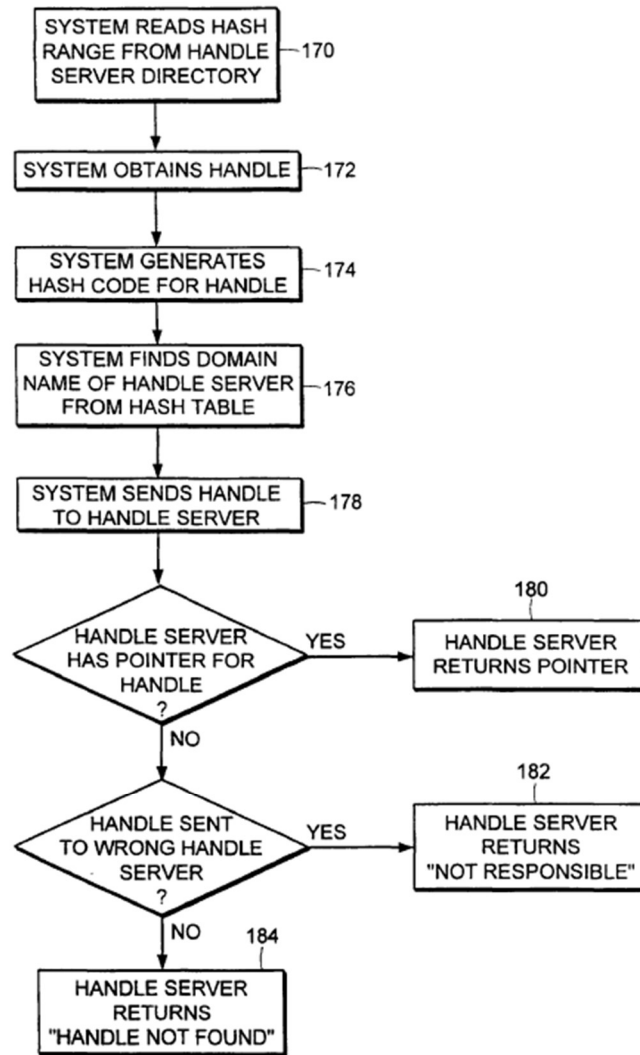
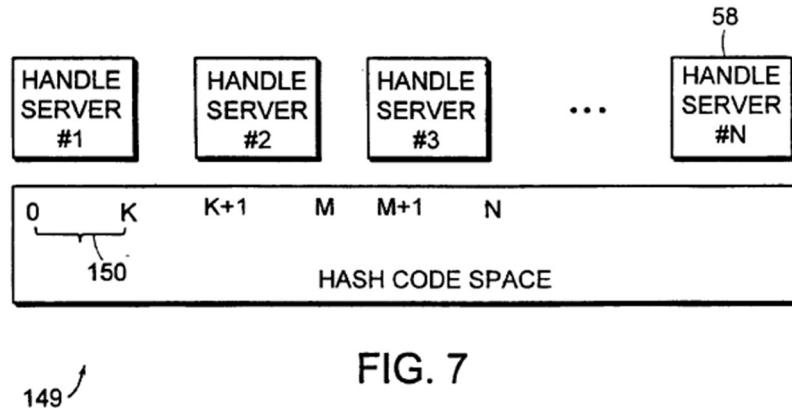


FIG. 8
SUBSTITUTE SHEET (RULE 26)

(*Id.*, FIG. 8.) In step 170, a client downloads from a handle directory server a hash range/handle-server-domain-name table (also referred to as a hash code/handle server table) that associates hash ranges with handle server domain names. (*Id.*, 46:22-47:5, FIG. 7.) The table allows the client to determine the location of a handle server that contains a pointer for retrieving the digital object associated with the handle. (*Id.*, 29:18-26, 38:9-17, 46:22-47:5 (“The system...consults the hash

range/handle-server-domain-name table to determine the domain name of the appropriate handle server (step 176).”), 73:4-74:26.)



(*Id.*, FIG. 7.) As shown in Figure 7, each handle server is assigned a range of hash codes/values. (*Id.*) The client generates a hash code for a handle using a predetermined hashing algorithm and then uses the hash code/value as an index into the downloaded table, allowing the client to obtain the domain name of the handle server that stores pointers for locating the digital object (“determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string”). (*Id.*, 47:1-11, 73:17-26; Ex. 1002, ¶63.)

Returning to Figure 8, after downloading the hash range/handle-server-domain-name table in step 170, the client in step 172 receives a handle from a user’s request (e.g., via bibliographic search or barcode scan) for obtaining a digital object. (Ex. 1005, 46:34-47:1, 73:8-16.) Subsequently, the client “generates the hash code for the handle using a predetermined hashing algorithm,”

then uses the hash code to index into the hash range/handle-server-domain-name table “to determine the domain name of the appropriate handle server,” and sends the handle to that server in steps 174, 176, and 178. (*Id.*, 47:1-11, 73:17-26; Ex. 1002, ¶64.)

Next, the handle server verifies if it does or, in the event of a dynamic change to the downloaded table, does not contain the handle and its associated pointers. (Ex. 1005, 47:10-11, 73:32-74:12.) If the handle is indeed located in the queried handle server, pointers for locating the digital object stored in the distributed network are returned to the client in step 180. (*Id.*, 29:8-10, 47:10-13, 73:23-26, 74:21-26.) The client then uses the retrieved pointers to obtain the digital object stored in a repository in the distributed network. (*Id.*, 74:21-29; Ex. 1002, ¶65.)

To the extent that Kahn does not explicitly disclose “**each one of the data location servers is configured to** determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string,” a POSITA would have been motivated and found it obvious to include this feature in view of Vingralek. (Ex. 1002, ¶66.)

Like Kahn, Vingralek relates to storing data in a distributed network. (Ex. 1002, ¶67.) Vingralek discloses a “distributed file organization system” where a file is “organized into buckets...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; Ex. 1002, ¶67.) Thus, Vingralek is similar to Kahn, which discloses a hash-based object distribution system and using a handle to locate/retrieve data. (Section IX.A.1.i.). A POSITA would have therefore been motivated to consider the teachings of Vingralek when implementing Kahn's system/method. (Ex. 1002, ¶67.)

Additionally, Vingralek teaches a server of its system can calculate the location of a correct server to forward a request to if the request was sent to a wrong server, i.e., a server that does not contain the requested information. As an initial matter, Vingralek discloses that 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. (Ex. 1007, 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 own 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, ¶68.)

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 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) would have gradually brought the client's perception of the address table up-to-date. Thus, Vingralek discloses that if a server does not have the requested information, each server ("each one of the data location servers") is configured to apply a hash function to the key ("based on the hash function applied to the identifier string") to determine another server to forward the request to ("to determine the at least one of the plurality of data location servers"). (Ex. 1002, ¶69.)

A POSITA would have been motivated to include features similar to those disclosed by Vingralek when implementing Kahn. Kahn teaches that every time a handle server does not contain the requested handle, a message is returned back to the client, triggering the client to download an updated hash table to determine/calculate the correct handle server, and the client subsequently redirects the request to the correct server. (Ex. 1005, 46:25-47:25, 73:1-74:32.) Vingralek's teaching limits communication overhead by having a server (e.g., the "wrong" server) directly forward a request to the correct server, thus limiting both additional communications to the client and additional downloads by the client. (Ex. 1007, 5,

8.) Vingralek's feature 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 limited computational resources (e.g., resources required for calculating the correct handle server). (Ex. 1002, ¶70; Ex. 1007, 5, 7, 8.) Such a modification would have been consistent with Kahn's goals of an "efficient and reliable implementation" in an expandable/scalable system where the server-client communications may increase (Ex. 1005, 9:23-27 ("handles used to uniquely identify digital objects are designed to be extensible and expandable to accommodate virtually any number of objects"))).

Moreover, while Kahn discloses that, when receiving a request, "[t]he handle server verifies that the handle falls within the set of handles for whose hash values it is responsible," it does not explicitly disclose the specifics of its verification method. (*Id.*, 73:32-34.) Thus, a POSITA would have been motivated to implement a verification method similar to that of Vingralek, i.e., a location server applying the hash function to determine which location server contains the requested information, as discussed above, especially because Kahn already discloses that the handle server stores a copy of the hash table. (*Id.*, 38:9-17, 46:29-32; Ex. 1002, ¶71.) Thus, a POSITA would have been motivated to include features similar to those disclosed by Vingralek, e.g., a server

determining/calculating the location of a correct server by applying the hash function to the handle and use the result as an index into the hash table for forwarding the request to the correct server, when implementing Kahn. (Ex. 1002, ¶71.)

A POSITA would have also had a reasonable expectation of success combining the teachings of Kahn and Vingralek. (Ex. 1002, ¶72.) As discussed above, both references are in the same field of art. (*See supra.*) Further, such an implementation would have involved modifying Kahn in a manner that Kahn suggests is feasible—the handle servers applying a hash function—as Kahn, like Vingralek, discloses that both client and handle server store copies of the hash table. (Ex. 1005, 38:9-17, 46:29-32.) Additionally, the implementation would have been consistent with Kahn’s teaching where a handle server sends a request to another handle server to “enhance the reliability and effectiveness of the system.” (*Id.*, 38:5-8.) Moreover, the combination of Kahn and Vingralek would have involved applications of known technologies (e.g., known distributed data storage systems (Kahn)) according to known methods (e.g., known methods where a server calculates the location of a correct server based on applying a hash function to an identifier for forwarding a request (Vingralek)) to yield the predictable result of a distributed system where a server calculates a correct server’s location for forwarding a request in a manner that improves the efficiency of the system, as

discussed above. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 416 (2007).

Accordingly, Kahn in view of Vingralek discloses and/or suggests this limitation.

2. Claim 2

Kahn discloses this limitation. (Ex. 1002, ¶73.) As discussed for limitation [1.b], Kahn discloses a hash algorithm (“hash function”) applied to a handle, with the output, i.e., a hash code/value, used to index into the hash range/handle-server-domain-name table (“hash table”). (Ex. 1005, 47:1-5; Section IX.A.1.iii.) Kahn further discloses all handles are partitioned/distributed amongst multiple handle servers based on the hash algorithm and hash range/handle-server-domain-name table, such that each handle server is assigned a portion of the handle space. (Ex. 1005, 7:5-9, 37:31-38:3, 46:22-24; Section IX.A.1.iii.) Since handles are stored with their associated pointers (“location information”) on handle servers, the associated pointers (“location information”) are likewise partitioned/distributed over multiple handle servers according to the hash range/handle-server-domain-name table, such that the pointers on a given server represent only a portion of the full set of pointers. A POSITA would have therefore understood the pointers (“location information”) on a given handle server comprise a portion of the full hash range/handle-server-domain-name table (“comprises any portion of a hash table”) because the handle server includes pointers associated with a portion of the handles in the hash range/handle-server-domain-name table. (Ex. 1002, ¶73.) A

POSITA would have also understood the pointers on a given handle server represent a portion of the hash range/handle-server-domain-name table generated by the hash algorithm (“generated with the hash function”) because the hash algorithm controls which handles are assigned to each handle server, and thus controls the associated pointers (“location information”) stored on each handle server. (*Id.*)

3. Claim 3

The Kahn-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶¶74-77.) To begin, Kahn discloses “wherein the data repository comprises a plurality of data repository servers.” For example, Kahn discloses a repository system (“data repository”) which may be comprised of multiple individual repositories. (Ex. 1005, 6:22-24.) Kahn discloses that each of these repositories stores information, performs services, and contains management systems associated with certain tasks, such as those associated with a repository access protocol for adding and accessing digital objects. (*Id.*, 11:16-27, 13:2-6, 21:3-5, 22:11-23.) Kahn further discloses that each of the repositories may store digital object properties records containing digital object metadata and transaction records containing entries regarding digital object transactions/requests. (*Id.*, 14:28-16:8.) Thus, a POSITA would have understood that each of the multiple repositories is a server (“a plurality of data repository servers”). (Ex. 1002, ¶74.)

The Kahn-Vingralek combination also discloses and/or suggests “wherein each of the plurality of data location servers is configured, in response to a request to store data in the data repository, to execute the hash function and store the data on one of the plurality of data repository servers based on a result of the hash function.” (Ex. 1002, ¶75.) Kahn discloses commands/requests to modify existing digital objects, such that new information is associated with an already existing digital object. (Ex. 1005, 19:26-31, 24:25-27.) A POSITA would have understood that such a request is a “request to store data in the data repository” which results in “stor[ing] the data on the plurality of data repository servers” because digital objects are stored in the data repository (and thus on data repository servers) and are comprised of sequences of bits/digits (“data”), and thus associating new information with an already existing digital object would have involved storing more sequences of bits/digits (“data”) with the digital object on the data repository servers. (Ex. 1002, ¶75.)

As discussed for limitation [1.b], in the Kahn-Vingralek combination, each handle server (“each of the plurality of data location servers”) is configured to execute a hash function to determine the handle server that stores pointers associated with the handle. (Section IX.A.1.iii.) A POSITA would have found it obvious to configure each handle server to also execute the hash function “in response to a request to store data in the data repository.” (Ex. 1002, ¶76.) A

POSITA would have understood that when modifying an existing digital object, it would have been advantageous to store the “data” corresponding to the digital object (and thus to the same handle) in similar locations (e.g., on the same data repository servers), so as to reduce processing requirements associated with the retrieval of a digital object. This is particularly true because, while Kahn discloses associating new information with an already existing digital object, Kahn does not expressly disclose the specifics regarding such a process. Thus, a POSITA would have found it obvious to implement the Kahn-Vingralek system such that, in response to a request to store additional “data” associated with a handle on the data repository servers (“in response to a request to store data in the data repository”), the system retrieves the pointers associated with the digital object’s handle to identify which repository servers should store the new data. *Id.* Consistent with the discussion for limitation [1.b], each handle server (“each of the plurality of data location servers”) is configured to execute the hash function (“to execute the hash function”) to determine the handle server that stores pointers associated with the handle. (Section IX.A.1.iii.) The system would then store the new “data” associated with an already existing digital object on one of the data repository servers (“one of the plurality of data repository servers”) based on the pointers associated with the handle at the particular handle server identified by the result of the hash function (“based on a result of the hash function”). (Ex. 1002, ¶76.)

A POSITA would have also had a reasonable expectation of success in implementing this modification because it would have involved modifying the Kahn-Vingralek system in a manner that Kahn suggests is feasible—implementing a method of modifying an existing digital object. (Ex. 1002, ¶77; Ex. 1005, 19:26-31, 24:25-27.) Such a modification would have involved applications of known technologies (e.g., known systems for storing and modifying digital objects (Kahn)) according to known methods (e.g., a server executing a hash function to retrieve location information (Vingralek)) to yield the predictable result of a system that has a handle server execute a hash function as part of the process to store data on a data repository server, as discussed above. *KSR*, 550 U.S. at 416. Accordingly, the Kahn-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶77.)

4. Claim 4

Kahn discloses this limitation. (Ex. 1002, ¶¶78-79.) Kahn discloses “a request for any location information associated with the respective one of the data entities.” For example, the client transmits a handle query in the form of a request pointer information UDP packet (“request”) to a handle server to retrieve pointers (“any location information”) associated with the digital object (“associated with the respective one of the data entities”) identified by the handle in the request. (Ex. 1005, 29:5-12, 47:5-11 (“The [client] system subsequently sends the handle to the

handle server as part of a request pointer information UDP packet...”), 73:32-34; *see also id.*, 7:5-7, 43:20-23, 44:6-7; Section IX.A.1.iii; Ex. 1002, ¶78.)

Additionally, Kahn discloses that “in response to” such a “request,” “each one of the data location servers is configured to return all location information associated with a respective one of the data entities.” For example, as discussed for limitation [1.b], Kahn discloses each handle server (“data location server”) returns all pointers (“all location information”) associated with a handle identifying a digital object (“data entity”) to the client. (Section IX.A.1.iii.) In particular, Kahn discloses “[a] handle server accepts a handle as input and returns a list of pointers associated with the handle,” where the handle identifies a digital object. (Ex. 1005, 29:8-12; *see* Section IX.A.1.iii.) Kahn further discloses that the full handle record (i.e., including the handle itself and all corresponding data) may be returned in response to a request from the client, which includes returning all pointers (“location information”) associated with a respective digital object (“respective one of the data entities”). (Ex. 1005, 42:1-11, 44:16-21; Ex. 1002, ¶79.)

5. Claim 5

Kahn discloses this limitation. (Ex. 1002, ¶¶80-81.) **First**, Kahn discloses “the hash table includes an association between a hash of the string identifier and at least one identifier of the at least one of the plurality of data location servers.”

As discussed for limitations [1.pre] and [1.b], Kahn discloses the hash range/handle-server-domain-name table (“hash table”) includes an association of a range of hash codes/values (“hash of the string identifier”)⁹ and the domain name of a handle server (“at least one identifier of the at least one of the plurality of data location servers”). (Sections IX.A.1.i, IX.A.1.iii.) Indeed, Kahn discloses calculating a hash code/value by applying a hash algorithm to the handle (“hash of the string identifier”) and using that value to search the hash table, which associates the hash of the handle (“a hash of the string identifier”) to the domain name (“at least one identifier”) of the handle server (“at least one of the plurality of data location servers”) containing the handle and its associated pointers. (Ex. 1005, 46:22-47:5, 73:4-74:26; Sections IX.A.1.i, IX.A.1.iii; Ex. 1002, ¶80.)

Second, Kahn discloses “wherein the location information includes a hash table.” As discussed for claim 2, the “location information” on a given handle server “comprises any portion of a hash table,” and thus a POSITA would have understood that the “location information” collectively comprises/includes a hash table. (Section IX.A.2.) To the extent claim 5 is read to require that the claimed “hash table” is “location information,” a POSITA would have understood that a

⁹ Petitioner assumes the claimed “string identifier” is intended to refer to the “identifier string” in claim 1. (Section IX.A.1.ii.)

hash table is “location information” because it provides information regarding the locations of handle servers containing requested pointers, and by extension the locations of digital objects (“data”) since the requested pointers indicate digital object locations. (Ex. 1002, ¶81.) Kahn also discloses that the hash range/handle-server-domain-name table is “location information” under the AWS Proceeding’s construction. (Section VIII.) As discussed above, the hash table provides information pertaining to the identities of one or more handle servers (i.e., location servers) because it maps hashes of handles to domain names (i.e., identities) of the handle servers. Additionally, the hash table provides information pertaining to one or more pointers (i.e., locations of data), which provide information about the location of a digital object (including “data”).

6. Claim 6

i) [6.pre]

To the extent the preamble of claim 6 is limiting, Kahn discloses the limitations therein. (Ex. 1002, ¶82.) As discussed for limitations [1.pre] and [1.b], Kahn discloses a system for managing pointers (“data location information”) and providing pointers in response to requests (“location queries”) to retrieve digital objects. (Sections IX.A.1.i, IX.A.1.iii.)

ii) [6.a]

Kahn discloses and/or suggests this limitation. (Ex. 1002, ¶¶83-88.) **First**, Kahn discloses “a location server configured to receive a location addition request...the location addition request comprising an identifier and at least one location to associate with the identifier.” Kahn discloses that when a digital object is deposited in a repository, it is made available in the handle system. (Ex. 1005, 20:4-14, 37:4-6.) In particular, Kahn discloses that when the digital object is deposited in a repository, a corresponding handle is generated and the handle and associated pointers are inserted into a handle server through an insertion request (“location addition request”). (*Id.*, 20:15-17, 39:34-40:1, 43:31-35, 45:10-21.) Thus, Kahn discloses that a handle server (“a location server”)¹⁰ receives a request to insert a handle and associated pointers (“location addition request”), which includes a handle (“identifier”)¹¹ and associated one or more pointers (“at least one location”) specifying the digital object’s location. (Ex. 1002, ¶83.)

¹⁰ Kahn discloses a “location server” as construed in the AWS Proceeding for the same reasons as discussed for limitation [1.b]. (Sections VIII, IX.A.1.iii.)

¹¹ Kahn discloses an “identifier” as construed in the AWS IPR for the same reasons as discussed for limitation [1.a]. (Sections VIII, IX.A.1.ii.)

To the extent Patent Owner argues or the Board finds that Kahn does not disclose a handle server receiving a location addition request that comprises a handle and associated pointers, a POSITA would have found it obvious to include such a feature in Kahn. (Ex. 1002, ¶84.) A POSITA would have found it beneficial to implement this feature because sending the handle (“identifier”) and associated pointers (“at least one location”) in a single request (“location addition request”) would allow for combining handle insertion and storage processes. Indeed, Kahn discloses processes for inserting handles and associated information (e.g., pointers) into a handle server (Ex. 1005, 20:4-17, 37:4-6, 39:34-40:1, 43:31-35, 45:10-21), and further discloses that the information is stored on the handle server in a table that associates handles with pointers (*id.*, 7:17-20, 37:31-33). A POSITA would have understood that combining the insertion and storage processes, such that already-associated handles and pointers are sent in a single location addition request, would have reduced processing steps and communication overhead since additional steps (beyond an insertion request) would not be required to associate handles and pointers on the handle server. (Ex. 1002, ¶84.)

A POSITA would have also had a reasonable expectation of success in implementing this modification. (Ex. 1002, ¶85.) Kahn discloses that handles and associated information may be sent to a handle server (Ex. 1005, 20:4-17, 37:4-6, 39:34-40:1, 43:31-35, 45:10-21), and thus the handle server would be readily able

to receive a location addition request containing a handle and associated pointers. Moreover, Kahn is already capable of storing handles along with associated pointers (*id.*, 7:17-20, 37:31-33). Such a modification would have involved applications of known technologies (e.g., known handle server storing a table that associates handles with pointers) according to known methods (e.g., known handle server receiving a request to insert handles and associated information) to yield the predictable result of a handle server receiving a handle and associated pointers as part of a location addition request, as discussed above. *KSR*, 550 U.S. at 416.

Second, Kahn discloses “the location addition request formatted in conformance with a transfer protocol.” Kahn discloses the handle system is based on the UDP protocol, which “enables a large number of transactions to be handled efficiently.” (Ex. 1005, 26:8-14, FIG. 33.) Indeed, Kahn discloses that messages/requests sent to a handle server are sent as UDP packets. (*Id.*, 47:5-8, 54:12-13, 59:23-25, 73:21-22.) Thus, a POSITA would have understood that transactions such as requests sent to a handle server to insert a handle and associated pointers are formatted in conformance with the UDP protocol (“a transfer protocol”). (Ex. 1002, ¶86.)

Third, Kahn discloses “wherein the identifier identifies an entity and wherein each of the at least one location specifies a location of data in a network pertaining to the entity.” As discussed for limitation [1.pre], Kahn discloses that a

handle identifies an object (“the identifier identifies an entity”) and a pointer indicates the location of the sequence of bits/digits (“data”) in the network pertaining to the object (“each of the at least one location specifies a location of data in a network pertaining to the entity.”) (Section IX.A.1.i; Ex. 1002, ¶87.)

Kahn discloses “data in a network pertaining to the entity” under the AWS Proceeding’s construction of “data pertaining to an entity.” (Section VIII.) Kahn discloses that its system is for managing information associated with real-world objects, including conventional works such as books, papers, images, and sounds (i.e., a thing that is real). (Ex. 1005, 1:16-20.) A POSITA would have understood that the sequences of bits/digits (“data”) pertaining to objects in the system are associated with real-world objects and form the data representation of those objects (i.e., digital objects), but are merely information stored in repositories and thus are distinct from the real-world objects themselves. (Ex. 1002, ¶88.)

iii) [6.b]

Kahn discloses this limitation. (Ex. 1002, ¶89.) As discussed for limitation [1.b], each handle server (“location server”) may be a computer and thus includes a “processor.” (Section IX.A.1.iii.)

iv) [6.c]

The Kahn-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶¶90-103.) **First**, Kahn discloses “programming logic stored on the

location server.” As discussed for limitation [6.b], Kahn discloses that each handle server (“location server”) may be a computer that includes a processor. (Section IX.A.6.iii.) Kahn further discloses that the handle system interfaces with the handle servers using “routines...written in the C programming language.” (Ex. 1005, 40:31-41:2.) Thus, a POSITA would have understood that operations/services provided on the computer, such as a handle-to-pointer translation service (*id.*, 21:6-31, 26:28-31, 29:5-12, 42:2-3, 44:6-7, 65:14-69:14), use associated “programming logic” stored and configured on each handle server to perform each of the operations of the handle server (Ex. 1002, ¶90).

Second, Kahn also discloses “wherein the programming logic is configured to return, in response to a location query related to a desired entity, a location message.” For example, Kahn discloses that a handle server (having “programming logic” stored and configured thereon) returns a message containing a list of pointers (“return[s]...a location message”) indicating the location of a digital object in response to a request related to an object (“in response to a location query related to a desired entity”). In particular, the request to retrieve the digital object includes sending the handle in the request to a handle server, whereupon a message containing a list of associated pointers is returned to the client to locate and retrieve the digital object. (Ex. 1005, 29:8-12, 47:5-11; Section IX.A.1.iii; Ex. 1002, ¶91.)

Third, Kahn discloses “the location message in conformance with the transfer protocol and comprising at least one location associated with the desired entity.” As discussed for limitation [6.a], Kahn discloses the handle system and associated transactions are based on the UDP protocol. (Ex. 1005, 26:8-14, FIG. 33; Section IX.A.6.ii.) Indeed, Kahn discloses that messages sent from a handle server, including messages containing pointers (e.g., as part of separate registration processes), are sent as UDP packets. (Ex. 1005, 59:28-31, 60:3-7, 60:15-18, 74:1-4, 74:13-17.) Thus, a POSITA would have understood that Kahn’s “location message[s]” are formatted in conformance with the UDP protocol (“a transfer protocol”). (Ex. 1002, ¶92; *see also* Ex. 1014, 1 (“[User Datagram Protocol] provides a procedure for application programs **to send messages** to other programs with a minimum of protocol mechanism.”), Ex. 1016, 4:37-5:12 (incorporating by reference RFC 768 [Ex. 1014]).) Further, the message containing a list of pointers (“location message”) includes at least one pointer and, as discussed for limitations [1.pre] and [1.b], the at least one pointer is associated with the desired object (“entity”). (Sections IX.A.1.i, IX.A.1.iii; Ex. 1005, 29:8-12, 47:10-11; Ex. 1002, ¶92.)

Fourth, Kahn discloses “wherein the programming logic is further configured to return the location message if the location server contains location information for the desired entity.” As discussed above, Kahn discloses returning

a message containing a list of pointers (“location message”) containing pointers (“location information”) indicating the location of the digital representation (i.e., digital object) of an object (“entity”). This message is sent if the handle server (“location server”) contains pointers associated with the desired object (“location information for the desired entity”). (Ex. 1005, 47:10-11; Ex. 1002, ¶93.)

Fifth, Kahn discloses “and wherein the programming logic is further configured to return a redirect message if the location server lacks the location information for the desired entity.” As an initial matter, in the present district court litigation, Patent Owner reads the claimed “redirect message” on a cache miss response due to stale or incorrect location information.¹² (Ex. 1019, ¶¶88-89 (“When the cached location is stale or incorrect, the client detects a miss, which is the redirect message....”).) Under Patent Owner’s interpretation, Kahn discloses this limitation, as explained below. (Ex. 1002, ¶94.)

For example, Kahn discloses transmitting a not-responsible-for-handle message (“redirect message”) to the client in the form of an invalid-handle-server-selected response UDP packet if the handle server, i.e., a “wrong handle server,”

¹² Petitioner does not acquiesce to Patent Owner’s interpretation of the “redirect message.” Nevertheless, Petitioner explains how Kahn discloses the “redirect message” under Patent Owner’s interpretation.

does not contain the handle and its associated pointers for an object (“if the location server lacks the location information for the desired entity”) indicating the location of the digital object. (Ex. 1005, 47:13-15 (“If the handle was sent to the wrong handle server, it returns a **not-responsible-for-handle message....**”), FIG. 8; *see also id.*, 73:1-74:26, 73:32-74:4 (“The handle server verifies that the handle falls within the set of handles for [which] it is responsible....**If it is not in this set, ...the handle server sends an invalid-handle-server-selected response UDP packet to the requestor....**”), FIG. 25.) Kahn discloses that the transmission of the not-responsible-for-handle-message to the client causes the client to download an updated table to determine/calculate the location of a correct handle server. (Ex. 1005, 47:13-21, 73:17-20, 74:5-12.) Thus, Kahn discloses a handle server (having “programming logic” stored and configured thereon) that returns a not-responsible-for-handle-message in the form of an invalid-handle-server-selected response UDP packet (“redirect message”) if the handle server (“location server”) lacks pointers (“location information”) indicating the location of information for the requested object. (Ex. 1002, ¶95.)

To the extent Kahn does not explicitly disclose “the redirect message **comprising a list of at least one other location server known to have the location information for the desired entity,**” a POSITA would have been

motivated and found it obvious to implement this feature in view of Vingralek.

(Ex. 1002, ¶96.)

As discussed for limitation [1.b], a POSITA would have been motivated to consider the teachings of Vingralek when implementing Kahn's system/method.

(Section IX.A.1.iii; Ex. 1002, ¶97.)

Vingralek additionally discloses propagating updates to clients' and other servers' address tables in a deferred manner. (Ex. 1007, 6.) As discussed for limitation [1.b], Vingralek discloses a method of deferred address table updates in which 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. (Section IX.A.1.iii.) In particular, 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. (*Id.*; Ex. 1002, ¶98.) Vingralek discloses that the "most salient feature" of its approach, where a client's address 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 storing data and reduce the cost of scaling up a distributed system. (Ex. 1007, 6, 8; Ex. 1002, ¶98.)

A POSITA would have been motivated to implement Vingralek's deferred address table updates in the Kahn system, for instance, by utilizing Kahn's existing mechanism of notifying the client. For instance, Kahn, similar to Vingralek, also uses a deferred update process, where the transmission of a not-responsible-for-handle message ("redirect message") results in the download of an updated hash range/handle-server-domain-name table for determining/calculating the location of a correct handle server after the initial attempt reached a wrong handle server. Thus, a POSITA would have been motivated to transmit an updated hash table to the client in such a deferred manner, i.e., in a not-responsible-for-handle message ("redirect message") after a wrong server was reached in an initial attempt, particularly given that Vingralek discloses such a method would limit communication overhead as discussed above. (Ex. 1002, ¶99.) Decoupling the communication overhead from the number of servers/clients in the system would have improved scalability, consistent with Kahn's goal and expectation of scaling. (Ex. 1005, 9:23-27, 33:3-14.)

A POSITA would have further understood that Vingralek's method of updating the client's address table via a redirect message would also be beneficial to provide the addresses of "correct" location servers to a client. (Ex. 1002, ¶100.)

Kahn discloses an arrangement in which a message is first returned to the client, upon which the client downloads an updated hash table, calculates/determines the location of a correct handle server, and forwards the request to the correct handle server. (Ex. 1005, 46:25-47:25, 73:1-74:32.) However, as discussed above, Vingralek discloses a method that both calculates the addresses of “correct” location servers, and updates the client’s address table, without requiring additional downloads or calculating a location server’s address by the client. A POSITA would have understood that Vingralek’s redirect message including an address table may be implemented such that only the portion of the address table corresponding to the location server(s) that contain the requested location information (“a list of at least one other location server known to have the location information for the desired entity”) would be sent in the message. This would allow the client to update the relevant portions of its own hash/address table, as well as provide the client with the addresses of the location server(s) that should receive the forwarded request. 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 handle server). (Ex. 1007, 5, 7, 8.) Thus, a

POSITA would have been motivated to include features similar to Vingralek's "redirect message" features when implementing Kahn. (Ex. 1002, ¶100.)

Consistent with the above and in such a combined system, for example, the client would apply the hash algorithm to the handle and its own copy of the hash range/handle-server-domain-name table and then send a request to a handle server ("location server"), which would apply the hash function to the handle and its own copy of the hash range/handle-server-domain-name table to determine whether it has the requested location information. (Ex. 1002, ¶101.) If the handle server does not have the requested pointers associated with the object ("if the location server lacks the location information for the desired entity"), the handle server would calculate the location of another handle server that has the requested pointers ("at least one other location server known to have the location information for the desired entity"). (*Id.*) The handle server would then send a redirect message containing a portion of the hash table corresponding to the handle servers known to have the pointers ("wherein the programming logic is further configured to return a redirect message...the redirect message comprising a list of at least one other location server known to have the location information for the desired entity"). (*Id.*) Such a redirect message (similar to Vingralek's disclosed message to update a client's address table) would allow the handle server to update its stale hash

table, and additionally provide the client with the addresses of a correct handle server, to which the handle server then forwards the request. (*Id.*)

A POSITA would have understood that the above-described implementation, in which the handle server calculates the location of a correct handle server for the client to forward a request to, would have been one of only several options (i.e., calculating and forwarding at either the client or the handle server that processes the information request). Thus, transmitting a redirect message with a list of at least one other handle server known to have the request pointers 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). A POSITA would have further understood that implementing such a redirect message would allow hash table updates to propagate from handle server to client, while reducing the number of downloads performed from the handle directory servers (i.e., by cutting down the number of downloads by the client). (Ex. 1002, ¶102.)

A POSITA would have also had a reasonable expectation of success combining the teachings of Kahn and Vingralek. (*Id.*, ¶103.) As discussed above, both references are in the same field of art. (Section IX.A.1.iii.) Moreover, Kahn itself suggests that its system/process is flexible and additional processes may be included, particularly with respect to the process of relaying a request between a

user and handle server. (Ex. 1005, 21 (explaining that “intermediary services” may be implemented between users and servers).) Additionally, it would have involved modifying the Kahn system in a manner that Kahn suggests is feasible—implementing a means of occasionally updating a hash range/handle-server-domain-name table, and implementing a means of providing domain names of correct handle servers to the client. (Ex. 1002, ¶103.) Moreover, such a modification would have involved applications of known technologies (e.g., known distributed system (Kahn)) according to known methods (e.g., known methods of providing deferred updated 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, Kahn in view of Vingralek discloses and/or suggests this limitation.

7. Claim 7

Kahn discloses this limitation. (Ex. 1002, ¶¶104-105.) As discussed for limitation [1.b], Kahn discloses that a hash table maps domain names of multiple handle servers in a network (“plurality of location servers in the network”) with corresponding hash codes/values (each a hash of a handle). (Section IX.A.1.iii; Ex. 1005, 38:9-17, 46:22-24.) As further discussed for limitation [1.b] with respect to the location information retrieval process, the hash table indicates which handle

servers store location information associated with a given handle. (Section IX.A.1.iii.) As discussed for limitations [1.pre] and [1.a], handles identify real-world objects (“plurality of entities”). (Sections IX.A.1.i-ii.) Thus, a POSITA would have understood that a hash table includes a list of handle servers in the network (“a list of a plurality of location servers in the network”), along with a corresponding list of hashed handles (which represent objects (“entities”)) that store location information on those servers (“corresponding list of a plurality of entities having location information on the respective one of the plurality of location servers”). (Ex. 1005, 7:8-9, 37:31-38:3, 46:22-24; Section IX.A.1.iii; Ex. 1002, ¶104.)

As discussed for limitation [6.c], a POSITA would have found it obvious to implement a redirect message returned to the client that contains a portion of the hash table corresponding to the “correct” handle server (“the list of at least one other location server”) that contains the requested pointers (“location information”). (Section IX.A.6.iv.) A POSITA would have understood that there may be multiple such “correct” handle servers, as Kahn discloses that data may be replicated across multiple handle servers. (Ex. 1005, 39:24-29; Ex. 1002, ¶105.) A POSITA would have thus understood that like the full hash table, the redirect message with a portion of a hash table (“the list of at least one other location server”) also comprises a list of the correct handle servers (“a list of a plurality of

location servers in the network”) and a corresponding list of hashed handles indicating which objects have location information on each server (“corresponding list of a plurality of entities having location information on the respective one of the plurality of location servers”). (Ex. 1002, ¶105.)

8. Claim 8

Kahn discloses this limitation. (Ex. 1002, ¶106.) As discussed for limitation [1.b], each of the plurality of handle servers (“location servers”) in the network stores only a portion of the pointers (“data location information”). (Section IX.A.1.iii.)

9. Claim 9

Kahn discloses this limitation. (Ex. 1002, ¶107.) Kahn discloses that pointers (“location information”) in a handle server (“location server”) are maintained in a table (“indexed location store”) associating handles with pointers. (Ex. 1005, 7:17-20, 37:31-33.) A POSITA would have understood that the handle server includes a storage medium to store the table (“indexed location store”), and that the table is an “indexed location store” because each handle is used as an index to retrieve the associated pointers (“location information”).

10. Claim 10

Kahn discloses this limitation. (Ex. 1002, ¶¶108-109.) First, Kahn discloses “the indexed location store comprises a string store.” Kahn explains that handles

are unique character strings (Ex. 1005, 7:4-5), and pointers consist of a “domain name of [the] storage system (repository)” (*id.*, 29:8-12), which a POSITA would understand is also a string. Thus, a POSITA would have understood that the handle server’s table associating handles with pointers comprises a “string store.” (Ex. 1002, ¶108.) Second, Kahn discloses that the handle server’s table (“indexed location store comprises a string store”) is “indexed by a hash table.” As discussed for claim 9, the handle is used as an index to retrieve all associated pointers from the handle server’s table. (Section IX.A.9.) Kahn discloses that the indices (i.e., the handles) of the handle server’s table are arranged/distributed to multiple handle servers based on a hash function/hash table (“indexed by a hash table”) that maps handles to handle servers. (Ex. 1005, 47:1-11, 73:17-27; Ex. 1002, ¶109.)

11. Claim 11

Kahn discloses this limitation. (Ex. 1002, ¶¶110-112.) As discussed for claim 10, Kahn discloses “the indexed location store comprises a string store indexed by a hash table.” (Section IX.A.10.) Kahn discloses multiple clients (“a plurality of clients”) (Ex. 1005, 38:34, 40:11), and explains that a hash range/handle-server-domain-name table (“hash table”) is shared/distributed to each client during initialization, and an updated table is shared/distributed to a client if the handle server does not contain the handle and its associated pointers (“a hash table distributed across a [client]”) (*id.*, 46:29-47:21, 73:4-74:12). Thus, a

POSITA would have understood that a hash range/handle-server-domain-name table may be shared/distributed across multiple clients (“a hash table distributed across a plurality of clients”). (Ex. 1002, ¶¶110-111.)

Kahn discloses the claimed “client” under the AWS Proceeding’s construction. (Section VIII; Ex. 1001, 4:14-17.) For example, Kahn discloses a user providing a handle to a client to obtain a pointer indicating the digital object’s network location. (Ex. 1005, 46:25-47:25, 73:1-74:32.) Moreover, Kahn discloses that the client’s user interface may be “in the form of software running on a workstation” (*id.*, 26:28-34, FIG. 2), and discloses that the user interface is part of a system that operates on a network (*id.*, 4:23-32, 19:26-31, 26:28-28:28, FIG. 2). Thus, a POSITA would have recognized that the client is a hardware and/or software component attached to a network. (Ex. 1002, ¶112.) As discussed for limitation [1.b], the client also sends a request (i.e., location request message) to a handle server (i.e., location server) to initiate lookup of mappings of handles (i.e., identifiers) to one or more pointers (which identify a digital object’s location). (Section IX.A.1.iii.)

12. Claim 12

Kahn discloses this limitation. (Ex. 1002, ¶¶113-114.) As discussed for claim 10, Kahn discloses “the indexed location store comprises a string store indexed by a hash table.” (Section IX.A.10.) Kahn discloses “a hash table

distributed across a plurality of servers.” For example, Kahn also explains the hash range/handle-server-domain-name table is distributed across handle directory servers (“a hash table distributed across a plurality of servers”), such that each handle directory server stores a hash range/handle-server-domain-name table. (Ex. 1005, 4:29-32, 29:18-26, 38:11-14, 44:8-10.)

13. Claim 13

The Kahn-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶¶115-116.) **First**, Kahn discloses “wherein the location query identifying the desired entity comprises a unique identifier for the desired entity.” As discussed for limitation [6.pre], Kahn discloses a request (“location query”) to retrieve a requested object (“desired entity”). (Section IX.A.6.i.) As discussed for limitation [6.c], the request sent from client to handle server includes the handle (“unique identifier”) for the requested object (“location query...comprises a unique identifier for the desired entity”), and thus a POSITA would have understood that the request identifies the desired object (“location query identifying the desired entity”) since each handle uniquely identifies an object. (Ex. 1005, 47:5-11; Section IX.A.6.iv; *see also* Section IX.A.1.ii; Ex. 1002, ¶115.)

Second, the Kahn-Vingralek combination discloses and/or suggests “wherein the programming logic stored on the location server further comprises programming logic configured to apply an index function to the unique identifier

to retrieve at least a portion of the locations associated with the unique identifier in the indexed location store.” (Ex. 1002, ¶116.) As discussed for limitation [6.c], Kahn discloses programming logic on a handle server (“location server”) that is configured to perform a handle-to-pointer translation service. (Section IX.A.6.iv.) As further discussed, in the Kahn-Vingralek combination, the handle server (“location server”) is configured to apply a hash algorithm (“index function”) to the handle (“unique identifier”) as part of the process for retrieving a portion of pointers (“at least a portion of the locations”) indicating the location of a digital object associated with the handle (“unique identifier”) that are stored in the table (“indexed location store”) of the handle server (“configured to apply an index function to the unique identifier to retrieve at least a portion of the locations associated with the unique identifier in the indexed location store”). (Section IX.A.6.iv.) A POSITA would have recognized that an index function is a function that returns an index to a table, and thus a hash algorithm/function, which returns a value that serves as an index to the hash table, is an “index function.” (Ex. 1002, ¶116.)

14. Claim 14

The Kahn-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶117.) As discussed for claim 10, Kahn discloses “the indexed location store comprises a string store indexed by a hash table.” (Section IX.A.10.) As

discussed for claim 13, the Kahn-Vingralek combination discloses and/or suggests that the hash algorithm applied to the handle is a hash function (“index function comprises a hash function”). (Section IX.A.13.)

15. Claim 15

i) [15.pre]

To the extent the preamble of claim 15 is limiting, Kahn discloses the limitations therein. (Ex. 1002, ¶118.) As discussed for limitation [1.pre], Kahn discloses a method for handling requests (“location queries”) in a network (“network”) to retrieve pointers identifying the location of a digital object, where the network contains multiple handle servers storing pointers (“a plurality of location servers including data location information”).¹³ (Section IX.A.1.i.)

ii) [15.a]

Kahn discloses this limitation. (Ex. 1002, ¶¶119-120.) As discussed for limitations [1.pre] and [1.b], Kahn discloses a network of handle servers (“the network”), and discloses that each handle server correlates/associates

¹³ Kahn discloses a “location server” and “plurality of location servers” as construed in the AWS Proceeding for the same reasons as discussed for limitation [1.b]. (Sections VIII, IX.A.1.iii.)

(“correlating”) each handle (“each one of a plurality of identifiers”)¹⁴ with one or more pointers indicating the location of the digital object in the network (“with at least one of a plurality of locations in the network”). (Sections IX.A.1.i, IX.A.1.iii.) As discussed for limitation [1.a], each handle (“each one of the plurality of identifiers”) uniquely identifies a digital object (“identifying a respective one of a plurality of data entities”), where the digital objects are stored in corresponding locations indicated by pointers, i.e., digital objects are stored in repositories, in the network (“wherein the data entities are stored in corresponding locations in the network”). (Section IX.A.1.ii; Ex. 1002, ¶¶119-120.)

iii) [15.b]

Kahn discloses this limitation. (Ex. 1002, ¶121.) As discussed for limitation [1.b], the client (“client”) transmits a request (“location query”) to a handle server (“receiving a location query from a client at one of the plurality of location servers”) for pointers (“the location query requesting location information”) associated with the handle in the request, where those pointers identify the location (“location information identifying a location”) of the specific digital object (among

¹⁴ Kahn discloses an “identifier” as construed in the AWS IPR for the same reasons as discussed for limitation [1.a]. (Sections VIII, IX.A.1.ii.)

the larger group of digital objects stored on the network) (“data entity included in the data entities”) associated with the handle. (Section IX.A.1.iii.)

iv) [15.c]

The Kahn-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶122.) As discussed for limitation [1.b], the Kahn-Vingralek combination discloses and/or suggests that each handle server may determine/calculate which other handle server (“determining which of the plurality of location servers”) contains the pointers (“includes the location information”) associated with a handle. (Section IX.A.1.iii.)

v) [15.d]

Kahn discloses this limitation. (Ex. 1002, ¶123.) As discussed for limitation [6.c], if a handle server determines that it has the requested pointers (“in response to determining the one of the plurality of location servers includes the location information”), it returns a message to the client (“sending a location response message to the client”) with the requested pointers (“the location response message comprising the location information”). (Section IX.A.6.iv.)

vi) [15.e]

The Kahn-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶124.) As discussed for limitation [6.c], the Kahn-Vingralek combination discloses and/or suggests that if a handle server determines it does not

have the requested pointers (“in response to determining the one of the plurality of location servers fails to include the location information”), it returns a message to the client (“sending a redirect message to the client”) identifying which of the other handle servers includes the requested pointers (“the redirect message identifying which of the plurality of location servers includes the location information”).

(Section IX.A.6.iv.)

16. Claim 16

Kahn discloses this limitation. (Ex. 1002, ¶125.) Kahn discloses that a handle identifying a digital object is hashed (“applying an identifier of the data entity”) using a predetermined hash algorithm (“to an indexing function”).

(Section IX.A.1.i.) The client then uses the output of the hash algorithm, i.e., a hash code/value, to index into the hash range/handle-server-domain-name table to determine the appropriate handle server (“to determine which of the location servers”) containing the pointers (“includes the location information”) specifying the digital object’s location. (*Id.*) If the handle server contains the handle and associated pointers, the pointers are returned to the client for it to locate and retrieve the digital object associated with the handle. (*Id.*; Ex. 1005, 47:1-13, 73:17-27.)

17. Claim 17

Kahn discloses this limitation. (Ex. 1002, ¶¶126-127.) **First**, Kahn discloses “wherein each one of the plurality of location servers comprises a portion of an indexed table of the identifiers and corresponding associated locations.” As discussed for limitation [1.b], Kahn describes a hash function that is applied to a handle (“identifier”) to generate an index to a hash table, where the hash table maps the hash of the handles (“identifiers”) to corresponding addresses of handle servers (“corresponding associated locations”). (Section IX.A.1.iii.) A POSITA would have understood that the hash table is therefore “an indexed table of the identifiers and corresponding associated locations.” As discussed for claim 2, Kahn discloses that each handle server (“each one of the plurality of location servers”) stores location information (i.e., pointers) that comprise a portion of a hash table. (Section IX.A.2.) Thus, each handle server (“each one of the plurality of location servers”) also comprises a portion of the hash table (“comprises a portion of an indexed table of the identifiers and corresponding associated locations”) because each server stores data corresponding to a portion of the hash table. (Ex. 1002, ¶126.)

Second, Kahn discloses “wherein determining which of the plurality of location servers includes the location information comprises applying one of the identifiers to a hash function to look up corresponding associated locations in the

indexed table, the one of the identifiers identifying the data entity.” As discussed for limitation [1.b], Kahn discloses determining the location of a handle server that stores the requested pointers (“determining which of the plurality of location servers includes the location information”) by using a hash function that takes a handle as an input (“applying one of the identifiers to a hash function”). (Section IX.A.1.iii.) As discussed above, the result of the hash function serves as an index for looking up associated handle server addresses (“to look up corresponding associated locations”) in the hash table (“in the indexed table”). As discussed for limitation [1.a], Kahn discloses that a handle (“identifier”) identifies the corresponding digital object (“data entity”). (Section IX.A.1.ii.) Thus, the handle in the request (“the one of the identifiers”) identifies the digital object that is ultimately retrieved (“identifying the data entity”). (Ex. 1002, ¶127.)

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.) *Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GMBH*, IPR2019-01469, Paper 6 at 8 (Feb. 13, 2020) (precedential). Moreover, there are material differences between the art/arguments here and those during prosecution

(*see, e.g.*, Ex. 1004, 94-101), including features believed to be missing (*id.*, 132-136). Discretionary denial is also not warranted in view of the completed *ex parte* reexamination proceeding, since this petition does not present the same or substantially the same prior art or arguments presented in that reexamination, and there are material differences between the art/arguments here with those presented therein (*see, e.g.*, Ex. 1037, 11-13), including features believed to be missing (Ex. 1039, 11-16.) 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. 1041; Section 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 Kahn, which is asserted in the only ground of the petition. (Ex. 1041, 13-19; Sections 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. (Sections II, IV; Ex. 1041, 19.)

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 (P.T.A.B. 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. *See Mueller* at 13.

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

Petitioner has never before filed a Patent Office challenge to the ’170 patent. Patent Owner may argue that the ’170 patent is already at issue in proceedings initiated by different parties. (Section 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 involving. *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 '170 patent, which "weighs especially heavily against a discretionary denial." See *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 (Section 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 '170 patent. *Unified* at 7-8. Petitioner has no relationship to the earlier challenges to the '170 patent. And there is no indication that AWS was aware of the Kahn-based prior art combination at issue here. In fact, 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. 1040, 217.)

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, 9.) The median time-to-trial in the Northern District of Illinois is close to 57 months. (Ex. 1035, 47.) The district court case was filed on June 30, 2023, so trial 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; *see also* 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 (P.T.A.B. 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,

Petition for *Inter Partes* Review
Patent No. 7,814,170

Dated: June 28, 2024

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

XII. APPENDIX A (CLAIM LISTING)

Claim 1

[1.pre] “A system for managing data stored in a distributed network, the system comprising:”

[1.a] “a data repository configured to store a data entity, wherein an identifier string identifies the data entity; and”

[1.b] “a data location server network comprising a plurality of data location servers, wherein data location information for a plurality of data entities is stored in the data location server network, at least one of the plurality of data location servers includes location information associated with the identifier string, each one of the plurality of data location servers comprises a processor and a portion of the data location information, the portion of the data location information included in a corresponding one of the data location servers is based on a hash function used to organize the data location information across the plurality of data location servers, and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string.”

Claim 2

“The system of claim 1, wherein the location information comprises any portion of a hash table generated with the hash function.”

Claim 3

“The system of claim 1, wherein the data repository comprises a plurality of data repository servers and wherein each of the plurality of data location servers is configured, in response to a request to store data in the data repository, to execute the hash function and store the data on one of the plurality of data repository servers based on a result of the hash function.”

Claim 4

“The system of claim 1, wherein each one of the data location servers is configured to return all location information associated with a respective one of the data entities in response to a request for any location information associated with the respective one of the data entities.”

Claim 5:

“The system of claim 1, wherein the location information includes a hash table and the hash table includes an association between a hash

of the string identifier and at least one identifier of the at least one of the plurality of data location servers.”

Claim 6

[6.pre] “A system for managing data location information and providing the data location information in response to location queries, the system comprising:”

[6.a] “a location server configured to receive a location addition request, the location addition request formatted in conformance with a transfer protocol, the location addition request comprising an identifier and at least one location to associate with the identifier, wherein the identifier identifies an entity and wherein each of the at least one location specifies a location of data in a network pertaining to the entity;”

[6.b] “wherein the location server includes a processor; and”

[6.c] “programming logic stored on the location server, wherein the programming logic is configured to return, in response to a location query related to a desired entity, a location message, the location message in conformance with the transfer protocol and comprising at least one location associated with the desired entity, wherein the

programming logic is further configured to return the location message if the location server contains location information for the desired entity, and wherein the programming logic is further configured to return a redirect message if the location server lacks the location information for the desired entity, the redirect message comprising a list of at least one other location server known to have the location information for the desired entity.”

Claim 7

“The system of claim 6, wherein the list of at least one other location server comprises a list of a plurality of location servers in the network and a corresponding list of a plurality of entities having location information on the respective one of the plurality of location servers.”

Claim 8

“The system of claim 6, wherein each of a plurality of location servers in the network stores only a portion of the data location information.”

Claim 9

“The system of claim 6, wherein the location information in the location server is maintained in an indexed location store.”

Claim 10

“The system of claim 9, wherein the indexed location store comprises a string store indexed by a hash table.”

Claim 11

“The system of claim 9, wherein the indexed location store comprises a string store indexed by a hash table distributed across a plurality of clients.”

Claim 12

“The system of claim 9, wherein the indexed location store comprises a string store indexed by a hash table distributed across a plurality of servers.”

Claim 13

“The system of claim 9, wherein the location query identifying the desired entity comprises a unique identifier for the desired entity, and wherein the programming logic stored on the location server further comprises programming logic configured to apply an index function to the unique identifier to retrieve at least a portion of the locations associated with the unique identifier in the indexed location store.”

Claim 14

“The system of claim 13, wherein the indexed location store comprises a string store and the index function comprises a hash function.”

Claim 15

[15.pre] “A method of handling location queries in a network, the network comprising a plurality of location servers including data location information, the method comprising:”

[15.a] “correlating each one of a plurality of identifiers with at least one of a plurality of locations in the network, each one of the plurality of identifiers identifying a respective one of a plurality of data entities, wherein the data entities are stored in corresponding locations in the network;”

[15.b] “receiving a location query from a client at one of the plurality of location servers, the location query requesting location information identifying a location of a data entity included in the data entities;”

[15.c] “determining which of the plurality of location servers includes the location information;”

[15.d] “sending a location response message to the client in response to determining the one of the plurality of location servers includes the location information, the location response message comprising the location information; and”

[15.e] “sending a redirect message to the client in response to determining the one of the plurality of location servers fails to include the location information, the redirect message identifying which of the plurality of location servers includes the location information.”

Claim 16

“The method of claim 15, further comprising applying an identifier of the data entity to an indexing function to determine which of the location servers includes the location information.”

Claim 17

“The method of claim 16, wherein each one of the plurality of location servers comprises a portion of an indexed table of the identifiers and corresponding associated locations, and wherein determining which of the plurality of location servers includes the location information comprises applying one of the identifiers to a

hash function to look up corresponding associated locations in the indexed table, the one of the identifiers identifying the data entity.”

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. 7,814,170 contains, as
measured by the word-processing system used to prepare this paper, 13,288 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. 7,814,170 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