

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

American Airlines, Inc., and
Hilton Domestic Operating
Company, Inc.,
Petitioners

v.

R2 Solutions LLC,
Patent Owner

Case IPR2023-00689

U.S. Patent No. 8,190,610

**Petition for *Inter Partes* Review of
U.S. Patent No. 8,190,610**

TABLE OF CONTENTS

STATUTES.....	10
RULES/REGULATIONS	10
OTHER.....	11
I. Introduction.....	12
II. Precise Relief Requested.....	12
A. Claims and Statutory Grounds	12
III. Overview of the Subject Matter.....	13
A. Overview of MapReduce	13
B. Overview of the '610 Patent	17
C. Prosecution History.....	19
D. The Level of Ordinary Skill in the Art.....	19
E. Claim Construction	19
IV. Overview of the Prior Art	24
A. Pike.....	24
B. Chowdhuri.....	26
A. Claim 1	29
[1a] “A method of processing data of a data set over a distributed system,”	29
[1b] “wherein the data set comprises a plurality of data groups,”	30
[1c] “the method comprising: partitioning the data...”	31
[1d] “providing each data partition to...”	32
[1e] “to independently output...”	33

[1f] “wherein the data of a first data group...”	37
[1g] “and the data of the first data group is mapped differently than...”	50
[1h] “wherein the different schema and corresponding different intermediate data...”	52
[1i] “reducing the intermediate data...”	53
[1j] “wherein the mapping and reducing operations are performed by a distributed system.”	55
B. Claim 17	56
[17b] “partitioning...”	56
[17c] “providing...”	56
[17d] “to independently output...”	56
[17e] “wherein the data...”	56
[17f] “and the data of the first data group...”	57
[17g] “wherein the different schema...”	57
[17h] “reduce...”	57
C. Claims 2 and 18 — “wherein: the at least one output data group is ...”	57
D. Claims 3 and 19 — “wherein: corresponding intermediate data for a data group...”	58
E. Claims 4 and 20 — “store[] [the corresponding intermediate data]...”	60
F. Claims 5 and 21 — “process[ing] the intermediate data... includes...”	60
G. Claims 6 and 23	62
[a] “wherein: the intermediate data includes...”	62
[b] “partition[ing] the intermediate data...”	62
H. Claims 7 and 24	63
[a] “the intermediate data includes...”	63

[b]	“partition[ing] the intermediate data...”	63
I.	Claims 8 and 25 — “wherein: in the partitioning step all of the key/value pairs of the intermediate data are provided to all of the partitions.”	66
J.	Claims 9 and 26	66
[a]	“wherein: at least some of the reducers include...”	66
[b]	“generating and providing metadata for...”	67
K.	Claims 10 and 27 — “wherein: the reducing [step] includes processing the metadata.”	68
L.	Claims 11 and 28 — “wherein: ...process[ing] the intermediate data for each data group in a manner that is defined to correspond to that data group includes...”	69
M.	Claims 12 and 22	69
[a]	“wherein: the intermediate data includes a plurality of grouped sets of key/value pairs” and reducing step via “a plurality of reducers,”	70
[b]	“for at least one of the reducers, the iterator corresponding to a particular data group, for that reducer, operates...”	70
N.	Claims 13 and 29 — “wherein: the intermediate data processing step of the reducing step further comprises...”	71
O.	Claims 14 and 30 — “...a plurality of reducers...” and “the data that is not intermediate data, for a particular reducer, includes...”	72
P.	Claims 15 and 31 — wherein reducing is carried out via “a plurality of reducers,” and “the data that is not intermediate data, for a particular reducer, includes data that is associated with that reducer.”	72
Q.	Claims 16 and 32 — reducing by “relating the data among the plurality of data groups.”	73
R.	Claim 33	74
[33a]	“A map-reduce...”	74
[33b]	“for a first data set...”	74

[33c] “partitioning...”	74
[33d] “providing...”	75
[33e] “to independently output...”	75
[33f] “for a second data set...”	75
[33g] “partitioning...”	75
[33h] “providing...”	76
[33i] “wherein the first schema differs from the second schema,”	77
[33j] “wherein the first and second schema...”	77
[33k] “reducing...”	78
[33l] “the output data set...”	78
[33m] “wherein the mapping and reducing operations...”	79
S. Claim 40	79
[40a] “A computer system...”	79
[40b] “for a first data set having a plurality of first key-value pairs”	79
[40c] “partitioning the first data set...”	79
[40d] “providing each data partition...”	80
[40e] “to independently output...”	80
[40f] “for a second data set”	80
[40g] “partitioning the first data set”	80
[40h] “providing each data partition...”	80
[40i] “wherein the first schema...”	80
[40j] “reducing the first and the second intermediate data set together...”	80
[40k] “and the output data set...”	81
[40l] “wherein the first and second schema and the first and second set of resulting	

key-value pairs...” 81

T. Claims 34 and 41 81

U. Claims 35 and 42 81

V. Claims 36 and 43 81

W. Claims 37 and 44 82

X. Claims 38 and 45 83

Y. Claims 39 and 46 83

VI. Mandatory Notices Under 37 C.F.R. § 42.8 83

A. Real Party-in-Interest 83

B. Related Matters 83

C. Lead and Backup Counsel.....85

D. Service Information.....86

VII. Grounds for Standing.....86

VIII. Conclusion 87

TABLE OF EXHIBITS

Exhibit	Description
EX-1001	U.S. Patent No. 8,190,610 to Dasdan (“the ’610 patent”)
EX-1002	U.S. Patent No. 7,590,620 to Pike
EX-1003	U.S. Patent Publication No. 2006/0218123 to Chowdhuri
EX-1004	File History of U.S. Patent No. 8,190,610
EX-1005	Declaration of Jimmy Lin, Ph.D.
EX-1006	Curriculum Vitae of Jimmy Lin, Ph.D.
EX-1007	Claim Construction Memorandum Opinion and Order, <i>R2 Solutions LLC v. Deezer S.A.</i> , 4:21-cv-00090, Dkt. 45 (E.D. Tex. Jan. 4, 2022).
EX-1008	Scheduling Order, <i>R2 Sols. LLC v. American Airlines Inc.</i> , No. 4:22-cv-00353, Dkt. 20 (E.D. Tex. Oct. 7, 2022)
EX-1009	Jeffrey Dean et al., MapReduce: Simplified Data Processing on Large Clusters, Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December 2004.
EX-1010	DocketNavigator Report for U.S. Patent No. 8,190,610 (last updated March 6, 2023)
EX-1011	Jeffrey Dean et al., MapReduce: Simplified Data Processing on Large Clusters, available at https://research.google.com/archive/mapreduce-osdi04-slides/index.html .
EX-1012	Plaintiff’s Disclosure of Asserted Claims and Infringement Contentions, 4:22-cv-00353 (E.D. Tex. Oct. 13, 2022)
EX-1013	Exhibit A of Plaintiff’s Disclosure of Asserted Claims and Infringement Contentions, 4:22-cv-00353 (E.D. Tex. Oct. 13, 2022)
EX-1014	Plaintiff’s Complaint for Patent Infringement, <i>R2 Sols. LLC v. Hilton Domestic Operating Company, Inc.</i> , No. 3:22-cv-02761, Dkt. 1 (N.D. Tex. Dec. 12, 2022)
EX-1015	[Reserved]

EX-1016	Kimberly Tripp, “SQL Server 2005: Partitioned Tables and Indexes,” SQLSkills.com (Sep. 24, 2004), <i>available at</i> https://www.sqlskills.com/resources/whitepapers/partitioning%20in%20sql%20server%202005%20beta%20ii.htm)
EX-1017	U.S. Patent No. 5,797,000 to Bhattacharya
EX-1018	U.S. Patent No. 7,020,661 to Cruanes
EX-1019	Plaintiff R2’s Opening Claim Construction Brief, R2 Sols. LLC v. American Airlines Inc, No. 4:22-cv-00353, Dkt. 42, (E.D. Tex. Feb. 21, 2023)
EX-1020	Oracle Database SQL Reference 10g
EX-1021	“Join (SQL),” Wikipedia.org (Aug. 11, 2004)
EX-1022	Hung-chih Yang, et al., <i>Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters</i> , SIGMOD ’07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (2007).
EX-1023	Notice of Docketing to the Court of Appeals to the Federal Circuit, <i>R2 Solutions LLC v. Deezer S.A.</i> , 4:21-cv-00090, Dkt. 45 (E.D. Tex. July 21, 2022).
EX-1024	U.S. Patent No. 6,343,295 to MacLeod
EX-1025	U.S. Patent Publication No. 2005/0165865 to Farmer
EX-1026	Listing of Claims of U.S. Patent No. 8,190,610

TABLE OF AUTHORITIES

CASES	PAGE(S)
<i>Alice Corp. Pty. v. CLS Bank Int’l</i> , 573 U.S. 208 (2014).....	8
<i>Apple v. Fintiv</i> , IPR2020-00019, Paper 15 (P.T.A.B. May 13, 2020).....	10
<i>In re GPAC Inc.</i> , 57 F.3d 1573 (Fed. Cir. 1995).....	8
<i>Netmut Ltd., v. Bright Data Ltd.</i> , IPR2021-01492, Paper 12 (P.T.A.B. Mar. 21, 2022).....	11
<i>Sand Revolution II, LLC v. Continental Intermodal Grp.-Trucking LLC</i> , IPR2019-01393, Paper 24 (P.T.A.B. June 16, 2020).....	10, 11, 12
<i>Sotera Wireless, Inc. v. Masimo Corp.</i> , IPR2020-01019, Paper 12 (P.T.A.B. Dec. 1, 2020).....	11
<i>Volkswagen Group of Am., Inc., v. Arigna Tech. Ltd.</i> , IPR2021-01321, Paper 10 (P.T.A.B. Feb. 15, 2022)	11
STATUTES	
35 U.S.C. § 103	2, 8
35 U.S.C. § 311	1
RULES/REGULATIONS	
37 C.F.R. § 42.8	75
37 C.F.R. § 42.24(a).....	79

OTHER

Memorandum from Director Vidal, Interim Procedure for Discretionary
Denials in AIA Post-Grant Proceedings with Parallel District Court
Litigation (USPTO June 21, 2022).....10

I. Introduction

The claims of the '610 patent recite a use case for MapReduce—a programming methodology that the patent admits was known. According to this use case, the input data to MapReduce is comprised of a plurality of “data groups,” two of which have different schema. But this was not new. The original developers of MapReduce disclosed their process, for example, in U.S. Patent No. 7,590,620 to Pike, and described mapping and reducing different types of input files, including documents from the Internet such as “text files, records, logs, sorted maps, etc.” This disclosure teaches, or at a minimum renders obvious, mapping and reducing data groups having different schema. U.S. Patent Publication No. 2006/0218123 to Chowdhuri also discloses this feature by applying iterators that correspond to Pike’s mapping and reducing operators to tables having different schema. The Board should institute review and cancel claims 1-46.

II. Precise Relief Requested

A. Claims and Statutory Grounds

Petitioners request review under 35 U.S.C. § 311 of '610 patent claims 1-46 and cancellation of those claims as unpatentable in view of:

Prior Art
U.S. Patent No. 7,590,620 to Pike (EX-1002); filed September 29, 2004; prior art under pre-AIA 35 U.S.C. § 102(e)
U.S. Publication No. 2006/0218123 to Chowdhuri (EX-1003); filed June 2, 2005, published September 25, 2006; prior art under pre-AIA 35 U.S.C. §§ 102(a), (e)

Grounds of Unpatentability	
1	Pike renders obvious claims 1-46 under 35 U.S.C. § 103 (pre-AIA).
2	Pike in view of Chowdhuri renders obvious claims 1-46 under 35 U.S.C. § 103 (pre-AIA).

III. Overview of the Subject Matter

A. Overview of MapReduce

In 2004, Google’s Jeffrey Dean and Sanjay Ghemawat published a programming methodology they called “MapReduce” for processing “parallel computations over distributed (typically, very large) data sets.” ’610 patent (EX-1001), 1:4-27; EX-1009; EX-1005 ¶¶35-37. MapReduce maps a set of data

into key-value pairs and reduces those key-value pairs into a list of values. '610 patent, 1:4-27. The mapping and reducing may be performed by different computers in parallel. *Id.*, 3:9-12. Figures 1 and 2 of the '610 patent depict the features of MapReduce. *Id.*, 2:9-3:8, Figs. 1, 2.¹

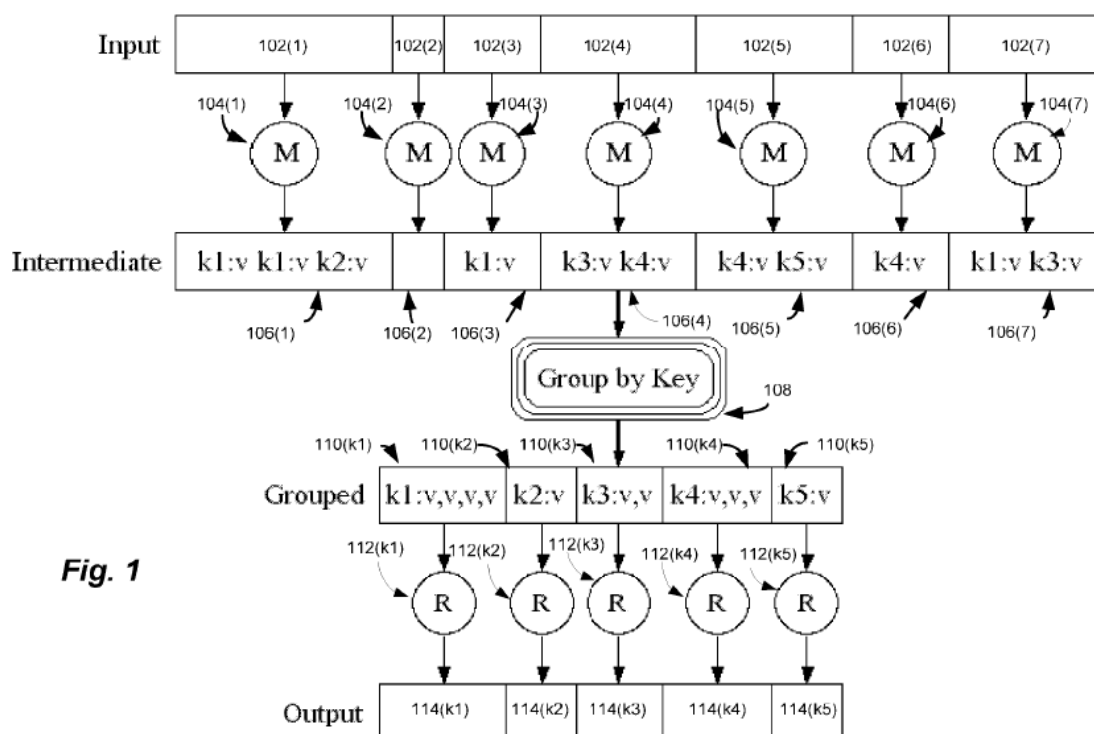


Fig. 1

Id., Fig. 1.

In Figure 1, input data set 102 comprises keys k1-k5. *Id.* 2:9-51. The input data 102 is divided into partitions 102(1)-102(7). *Id.* Each partition “is provided to[a] corresponding invocation of [a] map function,” 104(1)-104(7), to generate intermediate data 106(1)-106(7). *Id.* The intermediate data is then provided to

¹ Figures 1 and 2 are copied from a presentation by Dean and Ghemawat (EX-1011). '610 patent, 1:6-16.

corresponding reduce functions 112(k1)-112(k5). *Id.* Reduce functions generate corresponding output data 114, sorted by key 114(k1)-114(k5). *Id.* The map and reduce functions are application-specific, dependent on use case, and “user-provided.” *Id.*; EX-1005 ¶¶38-39.

Figure 2 (below) provides a parallel implementation of Figure 1, separated into three map tasks, 202a-202c, and two reduce tasks, 208a, 208b. ’610 patent, 2:52-67.

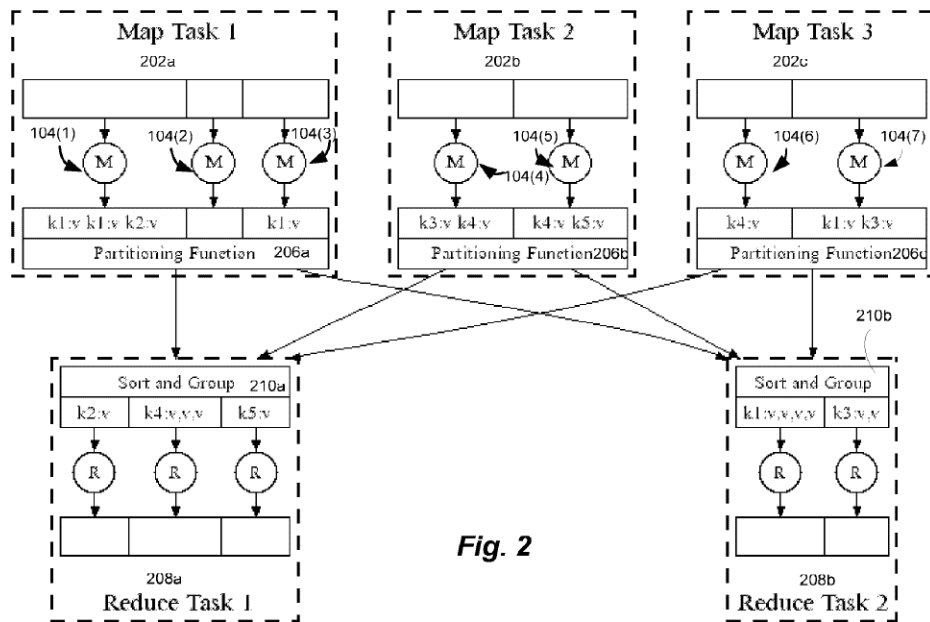


Fig. 2

Id., Fig. 2; EX-1005 ¶40.

The Dean and Ghemawat publication on MapReduce, cited by the ’610 patent, provides real-world uses of MapReduce. In one example, MapReduce is used to “count[] the number of occurrences of each word in a large collection of documents.” EX-1009, 2. The map function “emits each word plus an associated

count of occurrences,” while the reduce function “sums together all counts emitted for a particular word.” *Id.* While that example concerned a generic “collection of documents,” other examples included a different or specific type of input, namely, webpages: listing webpages that link to a target webpage (i.e., a “Reverse Web- Link Graph”), summarizing the most important words from different webpages (i.e., a “Term-Vector per Host”), and identifying webpages that include a specified word (i.e., an “Inverted Index”). *Id.*, 2-3; EX-1005 ¶¶41-42.

One of Google’s “most significant uses of MapReduce” by 2004 was indexing webpages for Google’s web search service. EX-1009, 10-11. The indexing process used “as input a large set of documents that have been retrieved by [Google’s] crawling system,...[t]he raw contents for these documents are more than 20 terabytes of data.” *Id.* Dean and Ghemawat’s examples demonstrate what a person of ordinary skill in the art (“POSITA”) already knew before the ’610 patent: that MapReduce could process diverse data sets and analyze large volumes of input data in an efficient manner. EX-1005 ¶44. As expected, the Dean and Ghemawat paper on MapReduce placed no limitations on MapReduce’s input data, including its type or format. *Id.* Instead, it contemplated operation on “large collection[s] of documents,” including websites and logs from the Internet that a POSITA would have known have different schema. *Id.*; EX-1009, 2, 10-11. It further discloses a MapReduce library providing “support for reading input data in several formats,” including a text mode that “treats each line [of a document] as a key/value pair.” *Id.*

at 6-7. A POSITA would have understood that there was no limitation on the type of data MapReduce could consume in 2006. EX-1005 ¶45.

B. Overview of the '610 Patent

The '610 patent purports to disclose “enhanced” MapReduce. '610 patent, Abstract, 1:29-44. But that enhancement merely characterizes MapReduce’s input data: “[t]he inventors have realized that, by treating an input data set as a plurality of grouped sets of key/value pairs, the utility of the MapReduce programming methodology may be enhanced.” *Id.*, 1:66-2:8, 1:29-41, 2:9-10, 3:16-18, 3:35-41, Abstract. According to the specification, “data sets within the same group are characterized by the same schema; and data sets within different groups are characterized by different schemas.” *Id.*, 3:48-57. The '610 patent thus specifies the data in each partition 102 of MapReduce shown in Figure 1. This was not new. EX-1005 ¶¶53-54; EX-1009, 2, 10-11.

When discussing whether MapReduce provided the ability to map and reduce this type of input data, the '610 patent only alleges that MapReduce could not “*efficiently* process data from heterogeneous sources,” *id.*, 3:9-20², and that it was “impractical”, *id.* EX-1005 ¶54. The specification concedes that its “related . . . datasets” are just like those of conventional MapReduce: “characterized by a common primary key.” *Compare* '610 patent, Abstract, 1:34-36, 2:2-4, 8:47-52,

² All emphases added by Petitioners unless otherwise indicated.

with *id.* Fig. 1, 2:12-18, 2:49-51; EX-1005 ¶55.

Figures 4 and 5 depict the purported “enhanced” MapReduce applied to an Employee table 302 and a Department table 304. ’610 patent, 3:65-4:22, 8:15-46, Figs. 4, 5. These figures are indistinguishable from “conventional” MapReduce in Figures 1 and 2. *Compare id.*, Figs. 4, 5, with, Figs. 1, 2. Grouping input data and describing each group as having a different schema (organization) creates no meaningful distinction. EX-1005 ¶56.

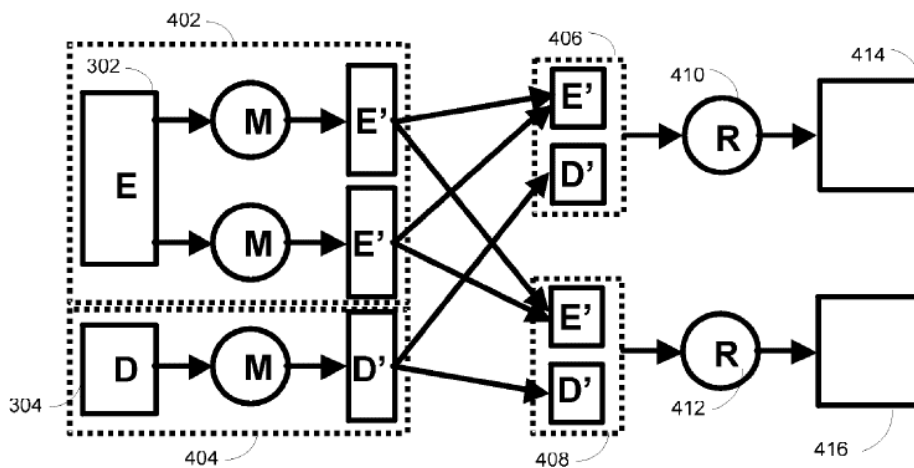


Fig. 4

Id., Fig. 4.

The inventors of the ’610 patent also acknowledged that MapReduce could process data “from multiple sources.” EX-1022, 1029-30. They noted that developers would only have to encode “an additional data-source attribute in the data and extra conditions in the code.” *Id.* Using these identifiers, “[u]ser-defined logic can extract data-sources from values to identify their origins, then the entries from different sources can be merged.” *Id.*, 1032. But encoding data tags or identifiers to

identify the source of data was a known MapReduce capability, and developers could easily encode input data this way before the '610 patent. EX-1005 ¶¶57. It was also known in the art. *Id.* ¶¶58-60; EX-1024, Abstract, 5:30-42, 5:53-60, Fig. 6; EX-1025, Abstract, ¶¶5, 8, 26-27.

C. Prosecution History

The Office rejected the application that would issue as the '610 patent three times under § 101, and four times under § 103. EX-1004, 202-239, 151-163, 109-123, 74-86. The Office withdrew its pre-*Alice* eligibility rejections when the applicant “clarif[ied] that the computer system comprises at least one processor and memory that are operable to perform such steps.” *Id.*, 13. The Examiner did not consider the art asserted in this Petition. '610 patent, Cover.

D. The Level of Ordinary Skill in the Art

Based on the relevant factors, *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995), a POSITA would have had a minimum of a bachelor's degree in computer science or similar field, and approximately two years of industry or academic experience in a field related performing data analytics and/or related data processing tasks. EX-1005 ¶¶64-66. Work experience can substitute for formal education and additional formal education can substitute for work experience. *Id.*

E. Claim Construction

In a case not involving Petitioners, the Eastern District of Texas construed several patent terms:

<u>Term</u>	<u>Construction</u>
a plurality of mapping functions that are each user-configurable	two or more mapping functions that are each configurable by a user
the data of the first data group is mapped differently than the data of the second data group	plain meaning
data set	plain meaning
data group	a group of data and a mechanism for identifying data from that group
data partition	plain meaning
including processing the intermediate data for each data group in a manner that is defined to correspond to that data group, so as to result in a merging of the corresponding different intermediate data based on the key in common	plain meaning
the at least one output data group is a plurality of output data groups	plain meaning

EX-1007, 13-31. The preambles of claims 1 and 17 were also found limiting. *Id.*, 19-23. For mapping the data groups “differently,” the court rejected arguments that the claims require different mapping *functions*. *Id.*, 16-18.

In the District Court dispute between Petitioner American Airlines, Inc. (“American Airlines Litigation”) and Patent Owner, Patent Owner and American Airlines, Inc. (“American”) are engaging in claim construction briefing at the time

of filing of this Petition (*see* EX-1019) and American³ has proposed the following constructions for the '610 patent:⁴

Claim Term	American Proposal
“identifiable to [that/the] data group”	indefinite
“independently output a plurality of lists of values”	“each map function outputs more than one list of values without coordination with other map functions”
“the data of the first data group is mapped differently than the data of the second data group”	“the data of the first data group is mapped by a different map function than the data of the second data group”

The Board need not expressly construe these terms, however, because the Petition shows that the '610 patent claims are invalid under any plausible construction, including American's proposed constructions.

³ In the District Court dispute between Petitioner Hilton Domestic Operating Company, Inc. and Patent Owner, the parties have not yet engaged in claim construction.

⁴ R2 asserts the '610 patent terms should be given their plain and ordinary meaning. Ex-1019.

F. The Board Should Institute *Inter Partes* Review

The Director’s interim procedure states that institution will not be denied based on the *Fintiv* factors “where a petition presents compelling evidence of unpatentability.” Memorandum from Director Vidal, Interim Procedure for Discretionary Denials in AIA Post-Grant Proceedings with Parallel District Court Litigation at 2 (USPTO June 21, 2022). The *Fintiv* factors need not be applied here because Petitioners have shown that claims 1-46 are unpatentable by a preponderance of the evidence.

Even if *Fintiv* applied, all of the factors either favor institution or are neutral. Factor 1 is neutral because neither party has requested a stay. *See, e.g., Sand Revolution II, LLC v. Continental Intermodal Grp.-Trucking LLC*, IPR2019-01393, Paper 24 at 7 (P.T.A.B. June 16, 2020) (informative). Factor 2 weighs against discretionary denial because no trial date is scheduled in either the American Airlines Litigation or R2’s suit against Petitioner Hilton Domestic Operating Company, Inc. (“Hilton Litigation”), thus any comparison to a projected final written decision would require speculation.⁵ EX-1008, 3. Even if trial is scheduled before the final written decision, the Board routinely institutes review. *Sand Revolution*, Paper 24 at 9-14 (five months before); *Netnut Ltd., v. Bright Data Ltd.*, IPR2021-

⁵ A scheduling order has not yet been entered in the Hilton Litigation.

01492, Paper 12 at 9-16 (P.T.A.B. Mar. 21, 2022) (six months before, no stipulation); *Volkswagen Group of Am., Inc., v. Arigna Tech. Ltd.*, IPR2021-01321, Paper 10 at 7-9 (P.T.A.B. Feb. 15, 2022) (three months before, no stipulation).

Factor 3 favors institution because the parties have expended limited resources in the litigation and Petitioners acted expeditiously by filing the Petition before the parties' *Markman* briefing, in the early stages of fact discovery, and before expert discovery in the American Airlines Litigation and within two months of the answer being filed in the Hilton Litigation. EX-1008, 1-2; *Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-01019, Paper 12 at 16-17 (P.T.A.B. Dec. 1, 2020) (precedential). Factor 4 favors institution because Petitioners challenge all claims of the '610 patent, although Patent Owner asserts a subset in the litigation. EX-1012, at 2 ("Claims 1–5 and 17–21"); EX-1014, 34 ("Claims 1–5 and 17–21"). Petitioners also stipulate that, if the Petition is instituted, Petitioners will not pursue the grounds identified in this Petition in litigation. *Sand Revolution*, Paper 24 at 11-12. Factor 5 favors institution because, while Petitioners are defendants in the parties' litigation, Patent Owner has asserted the '610 patent in twenty different litigations since January 2021, involving twenty-three different defendants. EX-1010, 1-2. Given Patent Owner's pattern of early settlement (with twenty-one defendants so far), and the patent's term extending to October 2029 ('610 patent, Cover), Patent Owner's campaign is likely to continue. Institution of review will promote judicial efficiency and integrity by reducing the number issues to be addressed in numerous parallel litigations.

Factor 6 favors institution because Petitioners have set forth a strong case of unpatentability of the challenged claims. *Sand Revolution*, Paper 24 at 13.

IV. Overview of the Prior Art

A. Pike

Pike, a patent assigned to Google on its face, names Jeffrey Dean and Sanjay Ghemawat as coinventors. Pike (EX-1002), Cover. Pike discloses a “large-scale data processing model 100” for MapReduce, which includes mapping and reduction operations in a map phase 102 and reduce phrase 104, respectively. *Id.*, 2:12-39.

In Figure 1, the mapping operations 102 are applied “to a set of input data α_i , (e.g., text files, records, logs, sorted maps, etc.) to provide a set of intermediate data values β_i .” *Id.* The reduction operations 104 reduce “the set of intermediate data values β_i , to provide a set of output data ϕ_i (e.g., tables, sorted maps, record I/O, etc.).” *Id.* Pike discloses that the mapping 102 and reducing 104 operations are “one or more application-specific” functions, *id.*, and that the data processing can be performed over distributed data processing system 500, *id.*, 2:41-3:35.

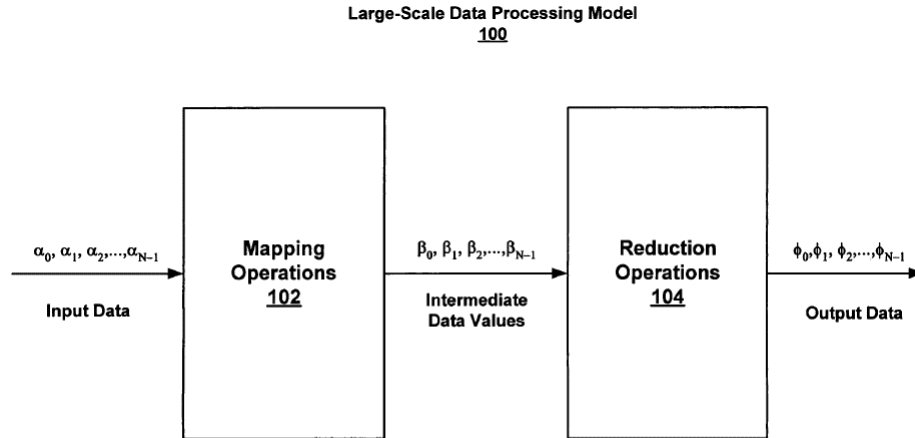
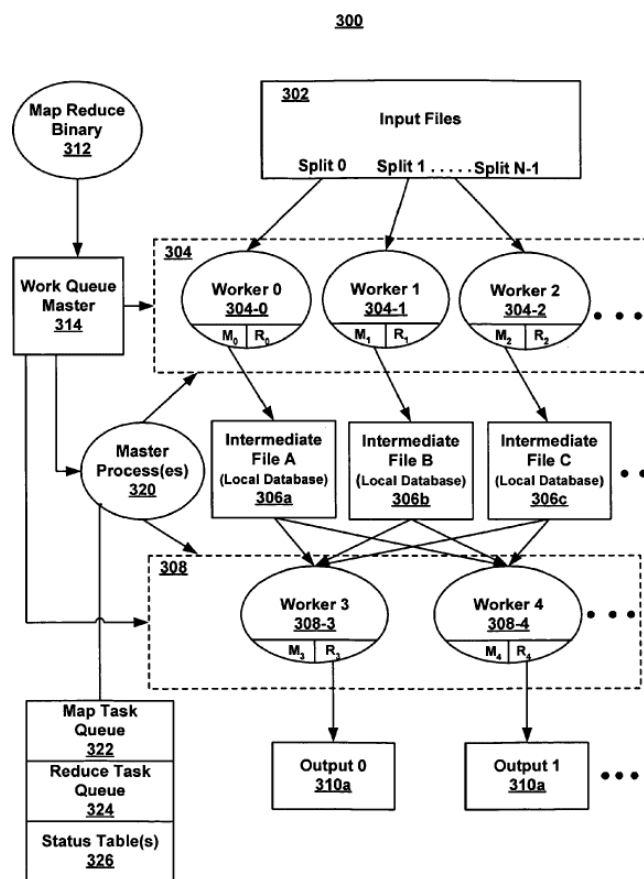


Figure 1

Id., Fig. 1.

In one embodiment, worker processes 304 and 308 invoke map and reduce processes, respectively. *Id.*, 8:65-10:27, Fig. 3. These operations can be carried out by different machines in system 300. *Id.* Inputs to the system include “input files 302,” which “can include a variety of data types typically used in data processing systems, including without limitation text files, record I/O, sorted data structures (such as B-trees), tables and the like.” *Id.*, 9:56-10:7. Pike discloses analyzing “groups of records” in another embodiment, which can be scanned and emitted during the map phase and include “ASCII, binary, or any other file type,” and even “unstructured” data. *Id.*, 15:62-21:6, 15:62-16:36, Fig. 9; EX-1005 ¶¶67-72.



Pike, Fig. 3.

B. Chowdhuri

Chowdhuri discloses methods for improving data query performance by dynamically partitioning data. Chowdhuri (EX-1003), Abstract. Chowdhuri employs iterators that are “self-contained software objects that accept a stream of rows from one, two, or multiple data sources.” *Id.* ¶¶129-30. In one embodiment, Chowdhuri discloses the use of its iterators in a “serial operator tree 300.” *Id.* ¶131, Fig. 3A. In this process, “order and customer tables (data 308, 309) are scanned by two separate scan iterators, 301 and 302, respectively. *Id.*

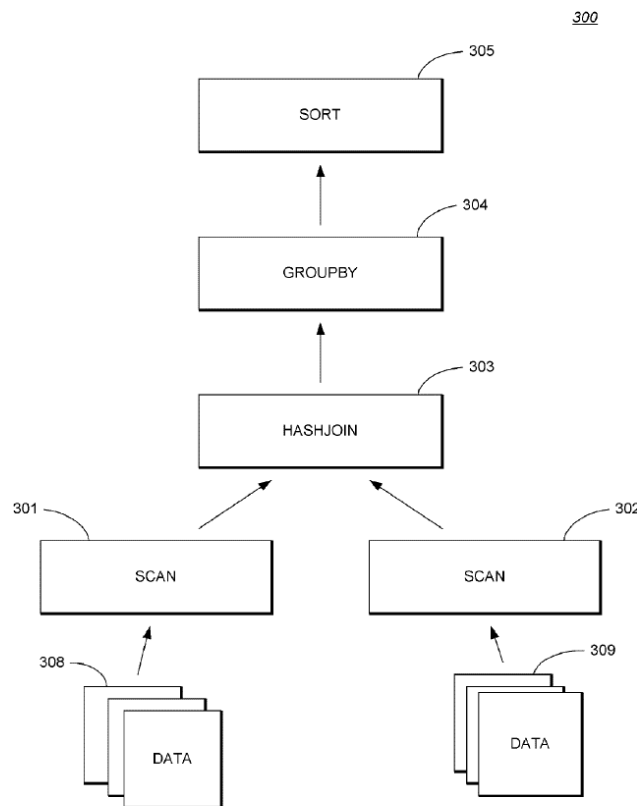


FIG. 3A

Id., Fig. 3A.

Chowdhuri’s scanning results are sent to a hash join iterator 303 to be matched, after which the “joined rows are then sent to the grouping iterator

(GroupBy) 304 that performs a grouping operation.” *Id.* ¶131. The GroupBy iterator 304 results are then sorted by the sort iterator 305. *Id.*

Chowdhuri also discloses using “exchange operators (iterators)” to “parallelize the serial query execution plan of FIG. 3A.” *Id.*, ¶135.

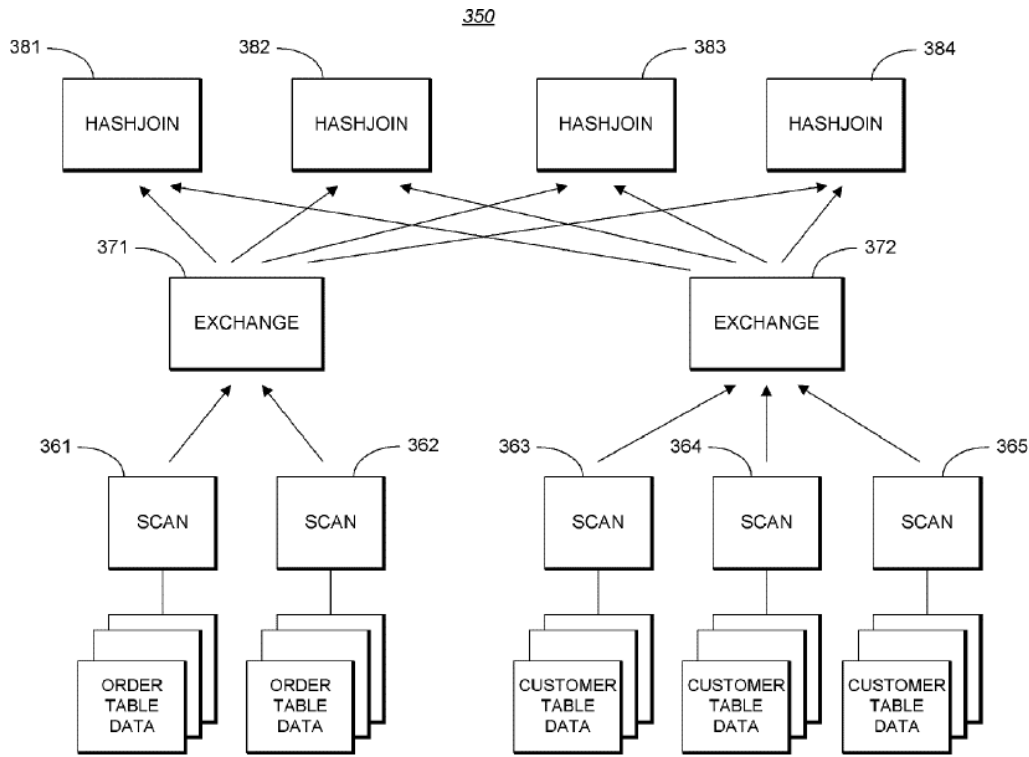


FIG. 3B

Id., Fig. 3B. The exchange operator “allows pipelining of intermediate results from one set of operators (i.e., producer tasks) and initiates another set of operators to process the next request (i.e., consumer tasks).” *Id.* ¶¶133-134. A POSITA would have understood that Chowdhuri’s iterators teach mapping and reducing, and performing these operations on tables having different schema. EX-1005 ¶¶74-78.

V. **Grounds 1 and 2: Pike, and Pike in View of Chowdhuri, Render Claims 1-46 Obvious**

A. **Claim 1**

[1a] “A method of processing data of a data set over a distributed system,”⁶

The '610 patent admits that this feature was “conventional.” '610 patent, 1:4-27, 3:9-15. Pike also discloses it by describing a method of “large-scale data processing” implemented “in a parallel and *distributed* processing environment.” Pike, 1:20-26, 2:14-39. For example, Pike discloses distributed data processing system 500 and computer system 400.

Data processing system 500 performs “large scale data processing,” and “the embodiments” of Pike. *Id.*, 2:41-3:35. Computer system 400 is used for “data processing systems 200 and 300,” and includes “mass storage that is *remotely* located from . . . central processing unit(s) 402.” *Id.*, 13:52-65, Fig. 4. The processes of large-scale data processing system 300 are distributed over a “number of machines,” and file reads are “distribut[ed] . . . over multiple local databases.” *Id.*, 9:20-28, 13:37-50.

A POSITA would have understood that Pike’s large-scale data processing system 300, operated over either computer system 400 or distributed processing

⁶ These headings correspond to the limitations in the listing of claims. EX-1026.

system 500, discloses processing data of a data set over a distributed system, as claimed. EX-1005 ¶¶80-83.

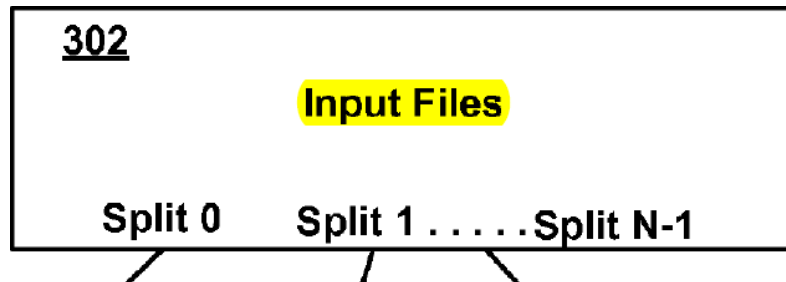
[1b] “wherein the data set comprises a plurality of data groups,”

The ’610 patent admits that this feature was conventional by disclosing that MapReduce operated on a data set comprising a plurality of groupings 102(1) through 102(7). ’610 patent, 2:9-35, Fig. 1. Each input data partition 102(1) through 102(7) is a grouping of data, or data group, that is provided to a corresponding invocation of the map function. *See id.*; EX-1005 ¶¶84-85.

Pike also discloses this feature. Pike’s input data for large-scale data processing system 300 comprises a plurality of data groups in “input files 302.” Pike, 9:58-64, Fig. 3. The input files 302 “include a variety of data types typically used in data processing systems, including without limitation text files, record I/O, sorted data structures (such as B-trees), tables and the like.” *Id.*, 10:1-4.

A POSITA would have understood that Pike’s input files 302 teach “data groups” because Pike’s input files are identified by, for example, a file name. EX-1005 ¶87. This corresponds with the ’610 patent’s disclosure of “groups of the original input data” as an “Employee table 302” and “Department table 304.” ’610 patent, 3:65-4:18, 7:18-23, Fig. 4; *see also id.*, 8:15-46, Fig. 5 (“Group: Emp” and “Group: Dept”). These input tables are data groups according to the patent because they are identifiable by file name or table name. EX-1005 ¶87. Indeed, a POSITA

would have understood that the '610 patent's tables would have been stored as files, like Pike's input files. *Id.*

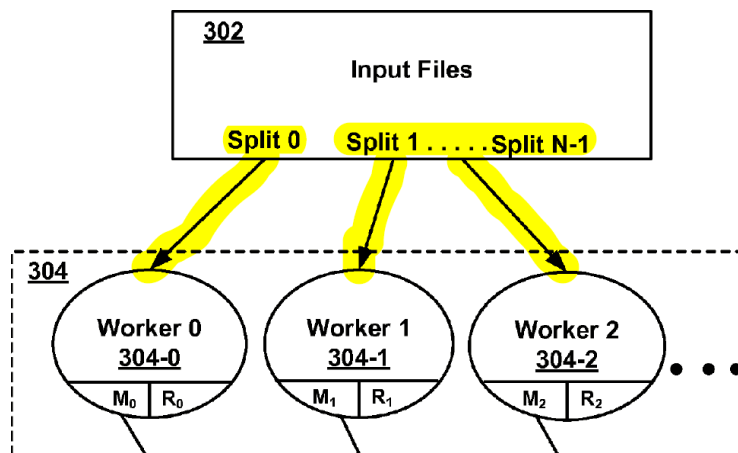


Pike, Fig. 3 (annotated).

[1c] “the method comprising: partitioning the data...”

The '610 patent admits that this feature was conventional by partitioning its input data before mapping. '610 patent, 2:19-35, Fig. 1.

Pike also discloses it. Pike discloses that its input files 302 are “[s]plit” via “[t]he division of input files 302 into data blocks 0, . . . , N-1.” Pike, 9:58-10:19, 14:57-63, Fig. 3 (below). Pike’s data blocks are then “treated as key/value pairs.” *Id.*,9:58-10:19.



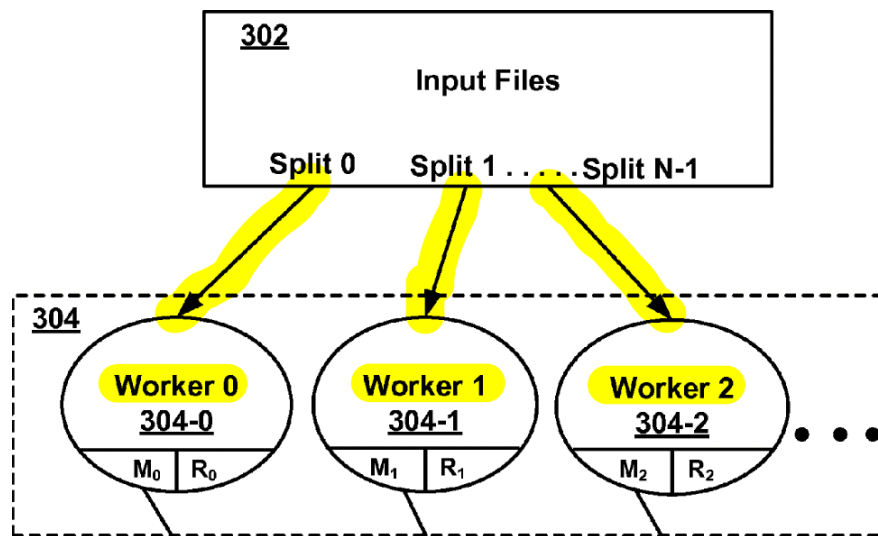
Id. at Fig. 3 (annotated).

Pike’s splitting input files into data blocks teaches the “partitioning,” as claimed. *Id.*, 9:58-10:19, 14:57-63, Fig. 3; EX-1005 ¶¶88-90. Pike’s “partitioning” is consistent with the ’610 patent, which discloses splitting its “Emp” and “Dept” tables into partitions (Step 502) and providing each partition to a mapping function (Step 504). ’610 patent, 8:15-45, Fig. 4; *see also id.*, 3:65-4:22, Fig. 4.

[1d] “providing each data partition to...”

The ’610 patent admits that this feature was conventional. ’610 patent, 2:33-35, Fig. 1.

Pike also teaches this feature by disclosing that its data blocks “are read by application independent worker processes 304-0, . . . , 304-N-1, as shown in FIG. 3.” Pike, 9:65-10:27, Fig. 3; *see also id.* at 14:57-63; EX-1005 ¶¶91-92.



Pike at Fig. 3 (annotated).

Pike discloses that “[e]ach of the worker processes 304 . . . applies [an] application-specific map() operator to the *respective* input data block so as generate

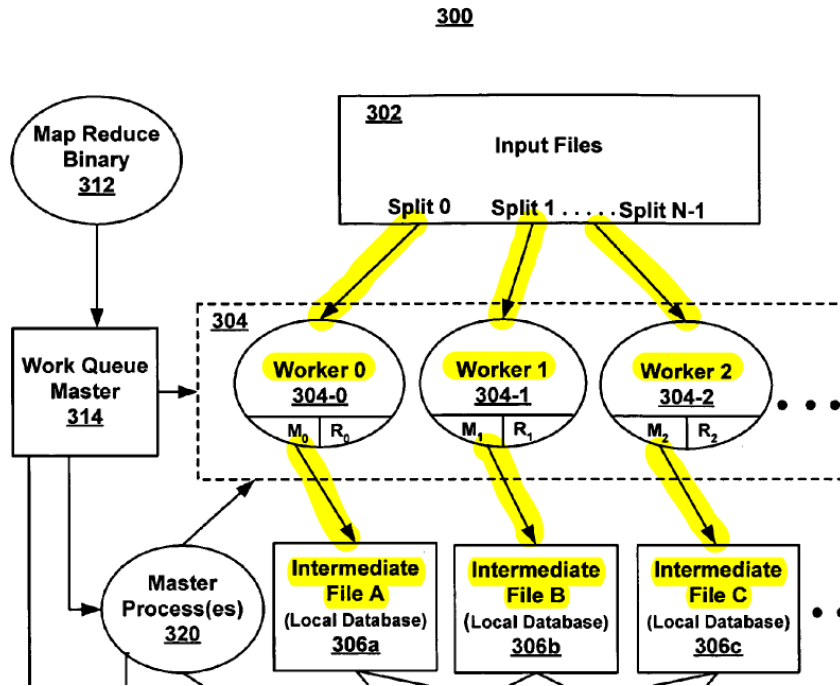
intermediate data values.” *Id.*, 9:65-10:27. The output is “collected and written to one or more intermediate files 306.” *Id.* An exemplary map operator “count[s] the number of occurrences of a term[] to provide intermediate data values.” *Id.*, 14:58-63, 10:36-39. Pike’s application of a map task to each “respective input data block” is consistent with the ’610 patent, which discloses portions of its tables being provided to specific mapping functions, and each mapping function providing intermediate data. ’610 patent, 8:15-24, Fig. 5; EX-1005 ¶¶93-94.

Pike also teaches that its map operators are “user-configurable.” Pike discloses that “an application programmer can create a script or program using the application software 422, which includes one or more [map, reduce, and partition] operators 424, 426 and 428.” Pike, 14:35-39, 14:12-27, 4:28-53, 5:26-39, 8:34-45; EX-1005 ¶95. Pike’s disclosure of map operators developed by application programmers teaches “two or more mapping functions that are each configurable by a user.” EX-1007, 13-15.

[1e] “to independently output...”

Pike discloses that each worker process 304 “invokes a map thread to handle an assigned map task.” Pike, 8:65-9:19, 10:4-27, 14:57-15:6. In one embodiment, “each of the intermediate files 306 receives output from **only one** worker process 304, as shown in FIG. 3.” *Id.*, 10:13-15, Fig. 3. Figure 3 depicts each worker process 304 independently outputting intermediate data to intermediate files 306 in Figure 3. *Id.*, Fig. 3 (below). A POSITA would have understood that this teaches map

functions “independently outputting” intermediate data because each worker process 304 invokes a map task to individually output intermediate data. EX-1005 ¶¶96-97.



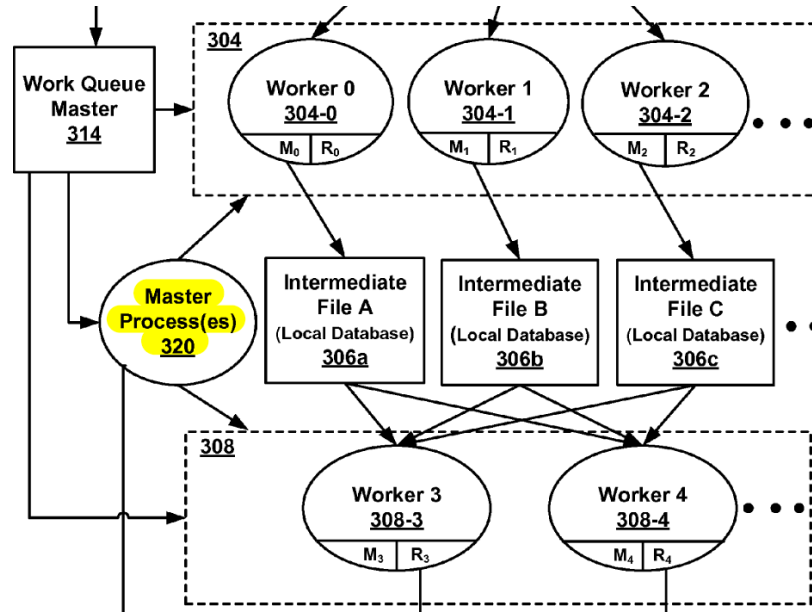
Pike, Fig. 3 (annotated).

Pike also discloses outputting “a plurality of lists of values for each of a set of keys found in such map function’s corresponding data partition,” as claimed. For example, Pike explains that “[t]he input data blocks 0, . . . , N-1, which may in some embodiments be treated as key/value pairs, are read by application independent worker processes 304-0, . . . , 304-N-1.” *Id.*, 9:65-10:19. The worker processes 304 “appl[y] [an] application-specific map() operator to the *respective input data block* so as [to] generate intermediate data values.” *Id.* Each data block, as shown above in Figure 3, is provided to a specific worker process 304 for application of a map operator. *Id.*, Fig. 3. In one example, Pike discloses “output[ting] a key/value pair

for every word *w* in every document in [a] collection, where the key/value pair is $\langle w, 1 \rangle$.” *Id.*, 10:31-11:4, 14:60-63. This teaches outputting lists of values for each of a set of keys found in a map function’s corresponding data partition, because it is outputting values based on a key for each “respective input data block.” EX-1005 ¶98.

Pike additionally explains that its map operators form “corresponding intermediate data,” as claimed, in intermediate files 306. Pike, 10:7-15; EX-1005 ¶99. These files are “retained (i.e., they are persistent) until the reduce phase completes.” Pike, 10:7-15.

Pike further discloses that the values stored in intermediate files 306 are “for [a] data group and identifiable to that data group” by disclosing a master process 320 that tracks the processing of each input file through mapping and reducing. *Id.*, 13:9-21, Fig. 3 (item 320); EX-1005 ¶100. Pike explains that “master process 320 is responsible for assigning tasks to the worker processes 304 and 308 and for tracking their status and output.” Pike, 13:9-21.



Id., Fig. 3 (annotated).

Pike’s master process 320 “maintains status tables 326 for managing tasks, as described with respect to FIGS. 7A and 7B.” *Id.*, 13:7-17, Fig. 7A. In Figure 7A, map and reduce tasks are tracked according to “Input files” (i.e., Pike’s “data groups”). *Id.*, Fig. 7A. Pike teaches that “the input files field may specify a portion of an input file (e.g., where the portion comprises a data block) to be processed by the task.” *Id.* 6:55-7:12. In the status table of Figure 7A (below), process P0000 (corresponding to Map task Map0000) is associated with input file 12340, whose status is “completed,” and process P0330 (corresponding to Reduce task Red0000) is associated with the same input file 12340. *Id.*, Fig. 7A.

Task Status Table					
Task ID	Status	Process	Input Files	Output Files	...
Map0000	Completed	P0000	12340	12341	...
Map0001	Completed	P0001	12344	12345	...
⋮	⋮				
Map0103	In Progress	P0010	12030	12031	...
Map0104	Failed	P0011	10101	12102	...
Map0103b	In Progress	P0010	12030	12031	...
⋮	⋮				
Red0000	In Progress	P0033	12340, ...	14000	...
Red0001	Waiting	-	-	-	...

Figure 7A

Id. (annotated).

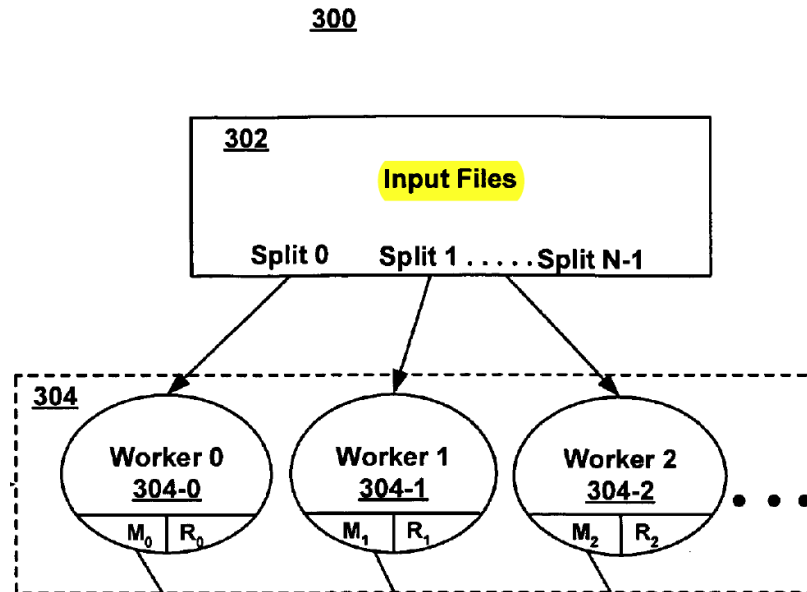
Pike’s tracking input files through the mapping and reducing processes, and generating intermediate data that is identifiable to a particular input file (e.g., input file “12340”), discloses outputting “intermediate data for [a] data group and identifiable to that data group.” EX-1005 ¶¶101-102. This is consistent with the ’610 patent, which describes tracking intermediate data associated with its Employee or Department tables. ’610 patent, 4:1-22, Fig. 4. The ’610 patent’s tables and Pike’s file names are both data groups whose data are tracked through the mapping and reducing phases. EX-1005 ¶102.

[1f] “wherein the data of a first data group...”

The ’610 patent acknowledges that “conventional MapReduce” could process data having “different schema,” but alleges that it only could not do so “efficiently” or “[p]ractical[ly].” ’610 patent, 3:15-20. This feature was therefore known in the

art, as confirmed by the patent. EX-1005 ¶¶103-105. The inventors confirmed this by acknowledging that MapReduce could process data “from multiple sources”—including heterogeneous sources having different schema—and track the same with a simple “data source attribute.” EX-1022, 1029-30. This confirms what a POSITA would have known: that the ’610 patent claims nothing more than a use-case of MapReduce that specifies particular data in different input data groups. EX-1005 ¶105.

Pike (**Ground 1**), and Pike in view of Chowdhuri (**Ground 2**), also render this feature obvious. Pike discloses that its “*input files 302* can include a variety of data types typically used in data processing systems, including *without limitation* text files, record I/O, sorted data structures (such as B-trees), tables and the like.” Pike, 10:4-7. Pike similarly discloses that MapReduce could handle a diverse set of “input data α_i (e.g., text files, records, logs, sorted maps, etc.).” *Id.*, 2:17-41. In one example, Pike processes “a large collection of documents.” *Id.*, 10:31-48. A POSITA would understand that Pike’s “input data” and “input files” can be heterogeneous and therefore, that they could include data groups comprising different schemas. EX-1005, ¶106. Pike then shows and describes those “input files 302,” (e.g., data groups comprising different schema) being processed (i.e., being partitioned and mapped by worker processes 304). *Id.*, 9:58-10:19, Fig. 3.



Id. at Fig. 3 (annotated). Pike’s input files 302 being split and provided to respective map operators therefore teaches, or at a minimum suggests, data groups having different schema. EX-1005 ¶¶106-107. Pike’s disclosure of processing a variety of data types is moreover consistent with Patent Owner’s assertion of claim 1. EX-1013, 37 (asserting “load[ing] . . . in a large number of file formats,” infringes).

To the extent Pike does not expressly disclose processing input data with different schemas, a POSITA would have found it obvious to implement Pike with input files of different schema (**Ground 1**) based at least on Pike’s disclosure of tailoring map and reduce operations to a specific user application. EX-1005 ¶¶107-108; Pike, 2:22-26, 2:31-36, 3:59-64. For example, Pike discloses that a MapReduce programmer need only know the identity of the input files, where to send output data, and “two application-specific data processing operators, hereinafter referred to as map() and reduce(),” to process input files, and therefore, could easily modify the

MapReduce functionality based on the desired application. Pike, 4:33-39, 6:1-12, 10:4-7, 10:39-50, 11:36-38, 14:19-27. A POSITA would have recognized that analyzing input files of a diverse schema, including type or format, would have been one known use-case and obvious implementation of Pike's system 300. EX-1005, ¶109.

A POSITA would have been motivated to make this modification at least in view of Pike disclosure or suggestion of applying its techniques to input files having different attributes. Pike, 10:4-7; EX-1005 ¶107. Implementing Pike with input files of different schema would have been nothing more than implementing system 300 in a manner Pike suggests: to analyze "a variety of data types," including "tables." Pike, 10:4-7; EX-1005 ¶107. A POSITA would have had a reasonable expectation of success in making this modification as it would have yielded predictable results: a system that analyzes input files of different schema, partitions those input files, and tracks the files through the map and reduce phases. EX-1005 ¶107.

For example, a POSITA would have recognized that Pike's disclosure of application-specific map and reduce operators, developed by programmers to be application dependent, also suggests map and reduce processing based on use-case. EX-1005 ¶108; Pike, 2:22-26, 2:31-36, 3:59-64. Indeed, Pike discloses that an MapReduce programmer need only know the identity of the input files, where to send output data, and "two application-specific data processing operators, hereinafter referred to as map() and reduce()," to process input files. Pike, 4:33-39,

6:1-12, 10:4-7, 10:39-50, 11:36-38, 14:19-27. A programmer therefore designs the map and reduce functions to operate in a particular way based on a given use case. EX-1005 ¶108.

Analyzing input files of a diverse schema, including type or format, would have been one known use-case and obvious implementation of Pike's system 300. EX-1005 ¶109. For example, a POSITA would have found it obvious to, for example, map a variety of different document types or formats to search for particular words, as Pike discloses. *Id.*; Pike, 10:31-11:4. This is the very nature of MapReduce and Pike's system 300: analyzing large, diverse collections of documents using application-specific functions. EX-1005 ¶109. To the extent Patent Owner asserts otherwise, there is no dispute that POSITAs knew about, were implementing, and were using MapReduce, as disclosed in Pike, to analyze large data sets well before the '610 patent and therefore, would have been motivated to modify Pike to do so. EX-1005 ¶109; *see* Sections III.A, III.B. Indeed, the named inventors conceded that MapReduce could analyze data of different schema by simply tracking input files. *See* Section V.C; EX-1022, 1029-30, 1032. POSITAs were also computer programmers with several years of experience, who wrote map and reduce functions as part of their jobs and knew how to customize these functions for different input files, including files of different schema. EX-1005 ¶109. It would have been obvious and expected for them to do so when confronted with a large collection of data. *Id.*

Mapping input files of different schema in Pike would have been routine activity for a POSITA. *Id.* ¶110. It would have involved nothing more than applying a POSITA’s knowledge of application-specific programming known in MapReduce to the disclosure of Pike’s map and reduce operators to analyze input files 302 of different schema, including type or format. *Id.* The predictable result would have been a system that would map a large collection of documents of different type or format to, for example, search and count particular words. *Id.* A POSITA would have been motivated to implement Pike’s mapping and reducing on input files having different schema, and to maintain a schema-based delineation of data groups in the mapping and reducing, because it would have allowed mapping and reducing large collections of diverse data sets. *Id.*

While the ’610 patent alleges an “efficien[cy]” or “[p]ractical[ality]” limitation in mapping and reducing these input files in MapReduce, ’610 patent, 3:15-20, there was no such limitation. EX-1005 ¶111. Nor is there one in Pike. *Id.* Nothing in the background section of the ’610 patent (’610 patent, 1:4-27), the admitted prior art (*id.*, 1:66-3:15), or the Dean paper (EX-1009) suggests MapReduce or Pike was so limited. Instead, each discloses using application-specific operators to map and reduce large collections of documents, including web pages scraped from the Internet. *See* Section III.A. A POSITA would have understood documents scraped from the Internet include different schema. EX-1005 ¶111.

It also would have been obvious to implement Pike's mapping and reducing input files of different schema given the options of limiting MapReduce to analyzing homogeneous records or analyzing files of diverse character. EX-1005 ¶112. It would have been obvious to implement Pike's mapping and reducing techniques on diverse input files given these options and a POSITA would have been motivated to do so to accommodate large collections of diverse data sets. *Id.*

Pike also suggests analyzing diverse data sets separate from large-scale data processing system 300. In Figure 9, Pike discloses "an exemplary method for analyzing data records" that teaches analyzing "groups of records 905," which may include data in "ASCII, binary, or any other file type," and even "unstructured" data. Pike, 15:62-21:6, 15:62-16:36, Fig. 9. "[D]uring the map phase," an emitter operator 814 extracts intermediate data from the groups of records 905 in the form of "zero or more values." *Id.*, 15:62-16:36; 16:62-17:15. Pike discloses that "the analysis of each record 920 is stateless and thus the analysis of each record "do[es] not depend on the contents of other records." *Id.*, 16:6-26. Analyzing groups of records that include unstructured, ASCII, binary, or other file types teaches or at least suggests analyzing diverse data records, including data having different schema. EX-1005 ¶113. A POSITA would have known that the data in these data groups would include various attributes and properties, and not the same format, type, or syntax. *Id.*

It would have been obvious to a POSITA to incorporate Pike's emitting key-

value pairs from data groups into Pike's large-scale data processing system 300. EX-1005 ¶114. In doing so, a POSITA would have understood that this data would be mapped and reduced in Pike's system 300, and tracked by file name via master process 320. *Id.* Combining these data records with system 300 would have been nothing more than incorporating one known technique of producing key-value pairs in process 900, with system 300, to yield the predictable result of mapping and reducing data groups having different schema. *Id.* A POSITA would have been motivated to make this modification to accommodate large collections of diverse data sets. EX-1005 ¶112.

Claim 1 does not recite mapping and reducing tables. It recites "data groups." However, it would have been obvious to a POSITA to modify Pike to process and join tables having different schema based on Pike's suggestion of input files having "a variety of data types," including "tables," Pike, 10:1-4, with "joins"—a known relational processing technique, EX-1005 ¶115, ¶¶46-52. This would have enabled the system to accommodate a more diverse set of input data and would have involved nothing more than combining Pike's known MapReduce processing with known methods of joins in RDBMS and SQL queries, which were common in the art of relational processing. *Id.* ¶115, ¶¶46-52.

Pike in view of Chowdhuri (**Ground 2**) also renders this feature obvious. Chowdhuri discloses performing operations on data groups having different schema: order and customer tables. Chowdhuri, ¶¶127, 131, 135, Figs. 3A, 3B.

A POSITA would have understood Chowdhuri's order and customer tables are data groups having different schema because they are tables with different attributes—order data and customer data. EX-1005 ¶123. This is consistent with the '610 patent's employee and department tables. '610 patent, 3:23-34, 3:65-4:13, Fig. 3. A POSITA would have understood that Chowdhuri's disclosure of processing data groups having different schema would have been applicable to Pike's system, as each reference discloses similar data processing operations. EX-1005, ¶¶116-117.

For example, Chowdhuri discloses a system for “parallel query optimization using semantic- based partitioning.” Chowdhuri, Abstract. Chowdhuri discloses using a series of iterators to execute a data query, which Chowdhuri explains are “self-contained software objects that accept a stream of rows from one, two, or multiple data sources.” Chowdhuri, ¶¶131-136, Figs. 3A, 3B. Chowdhuri's iterators “do not know what the source of a data stream is; whether it is the contents of a disk file, a network connection, or even another iterator.” *Id.* ¶131, ¶61 (iterators “open data streams, iteratively read the streams, process it and close the streams”). A POSITA would have understood that Chowdhuri's iterators disclose techniques that correspond to Pike's mapping and reducing. EX-1005 ¶¶116-117. Specifically, a POSITA would have understood that Chowdhuri's scanning iterators correspond to Pike's map operator, while the hash join, GroupBy, and sort iterators correspond to Pike's reduce operator. *Id.*

For example, Chowdhuri's Figure 3A (below) teaches executing a data query

using a series of iterators to scan an order table 308 and a customer table 309. Chowdhuri ¶¶126-131, Fig. 3A. Chowdhuri’s tables “are scanned by two separate scan iterators, 301 and 302, respectively.” *Id.* ¶131. The table scan results “are sent to the hash join iterator 303 to be matched.” *Id.* The joined data is then grouped by a GroupBy iterator 304, and sorted by a sort iterator 305. *Id.*

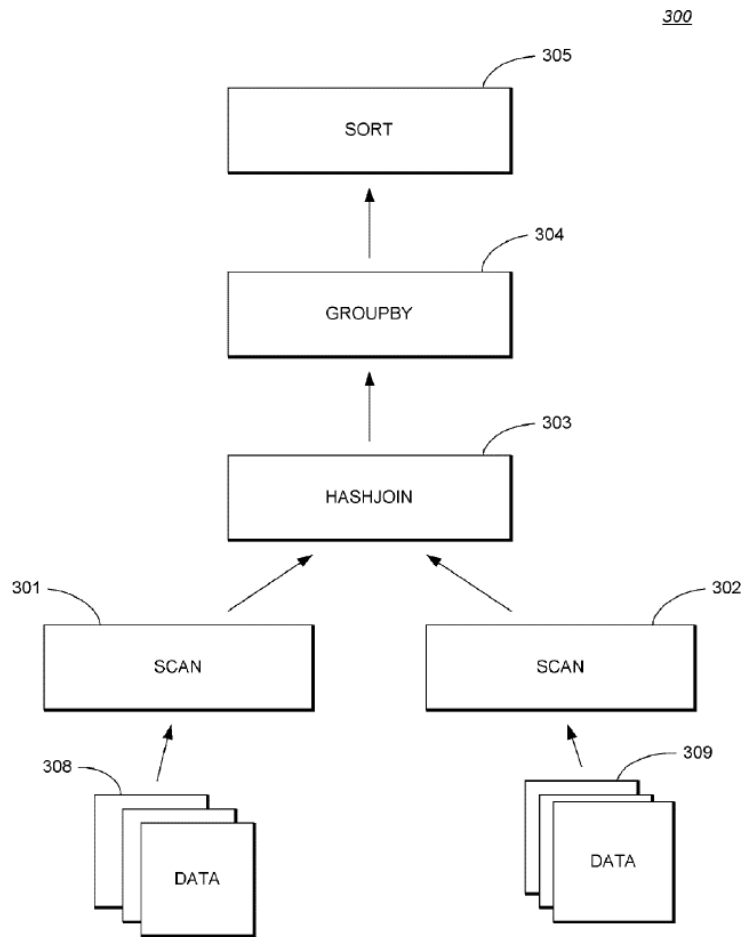


FIG. 3A

Id., Fig. 3A.

In Figure 3B (below), an “order table (data) is scanned in parallel by two threads, 361 and 362, while the customer table is scanned in parallel by three threads 363, 364, and 365, respectively.” *Id.* ¶135, Fig. 3B. After the scanning iterators 361-

365 process the tables, the results are provided to exchange operators, which “allow[] pipelining of intermediate results from one set of operators (i.e., producer tasks) and initiate[] another set of operators to process the next request (i.e., consumer tasks). *Id.* ¶ 134. The GroupBy iterator 304 and sort iterator 305 may then be applied after the hash joins in process 350 in Figure 3B. *Id.* ¶135.

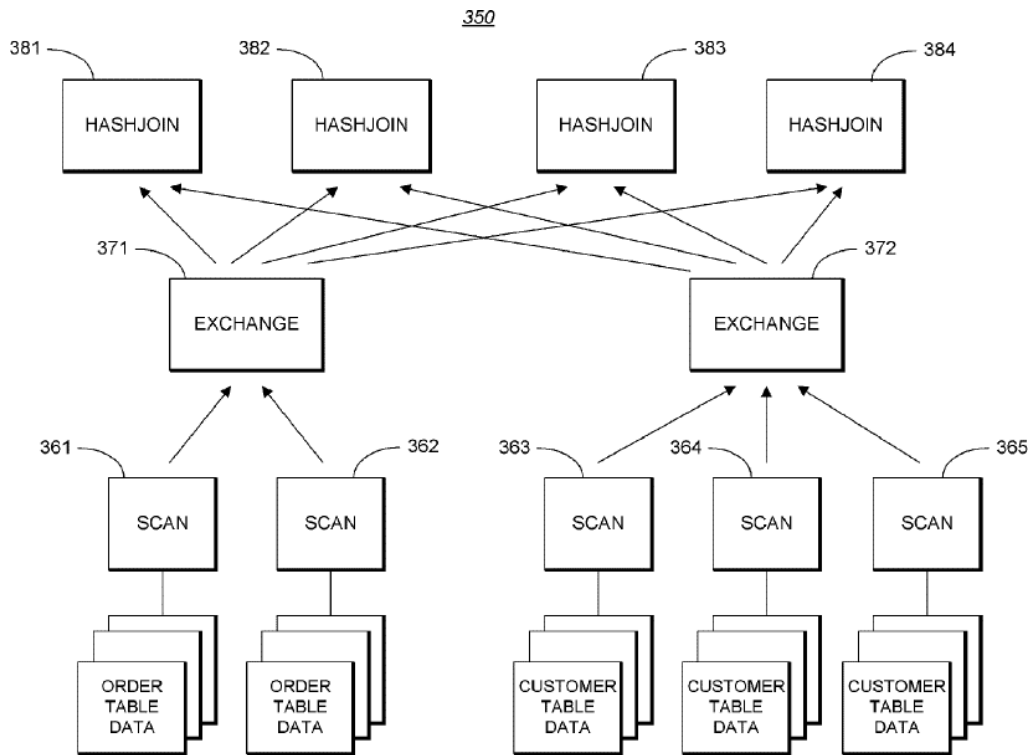


FIG. 3B

Id., Fig. 3B.

A POSITA would have understood that Chowdhuri discloses mapping input data to produce intermediate results via scan iterators 301 and 302 in Figure 3A, and scan iterators 361, 362, 363, 364, and 365 in Figure 3B. *Id.* ¶¶130-135, Figs. 3A, 3B; EX-1005 ¶¶118-120. Chowdhuri’s scanning iterators produce intermediate data as “results”: “[o]nce the[] tables are scanned, results are sent to the hash join iterator

303 to be matched.” Chowdhuri ¶¶131, 135; EX-1005 ¶120.

A POSITA would also have recognized that Chowdhuri’s hash join, GroupBy, and sort iterators reduce the resulting intermediate data from the scan iterators. Chowdhuri ¶¶127, 130, Figs. 3A, 3B; EX-1005 ¶121. The hash join iterator “get[s] the orders that belong to a particular customer,” the GroupBy iterator aggregates “orders by state [within the United States] and sums the order amount for each state,” and the sort iterator “implement[s] [a] sort enforcer and merge . . . used to implement a merge join or a merge union.” Chowdhuri ¶¶127, 130.

Chowdhuri’s iterators also correspond to the ’610 patent’s description of the map and reduce functions in MapReduce. ’610 patent, 1:20-27 (“[t]he map function iterates” and “[t]he reduce function . . . combines elements”); EX-1005 ¶122. They also correspond to how Dean and Ghemawat describe MapReduce’s reducing. EX-1009, 4. Chowdhuri’s scan iterator “iterates” over the input data to produce intermediate results, while the hash join, GroupBy, and sort iterators “combine[] elements” from the intermediate results. EX-1005 ¶122. Hash join, GroupBy, and sort iterators are also expressly contemplated by the ’610 patent. ’610 patent, 5:19-39 (sorting), 2:65-66 (“a sort and group-by-key task”), 5:48-6:7 (group iterator), 6:34-7:16 (“hashtable” iterators). They are also claimed: “the reducers include a sort, group-by-key and combine task.” *Id.*, cl. 9. The sort iterator further corresponds to the reducers in Pike. Pike, 11:28-31.

A POSITA would have been motivated to apply the teachings of Chowdhuri to Pike's system 300 with a reasonable expectation of success. EX-1005 ¶124. A POSITA would have implemented Chowdhuri's scanning iterator as mapping operator in Pike, and one or more of Chowdhuri's hash join, GroupBy, or sort iterators as Pike's reducing operator, as Pike suggests by its sorting intermediate data values during the reducing. *Id.*; Pike, 11:28-31. In Chowdhuri-Pike system, Chowdhuri's order table and customer table would be partitioned and mapped in system 300 to output intermediate data identifiable to these tables, and reduced in system 300, for example, by a "state" within the United States. Chowdhuri ¶¶126-27 (grouping data from tables "by state."), ¶131 (grouping by "state codes").

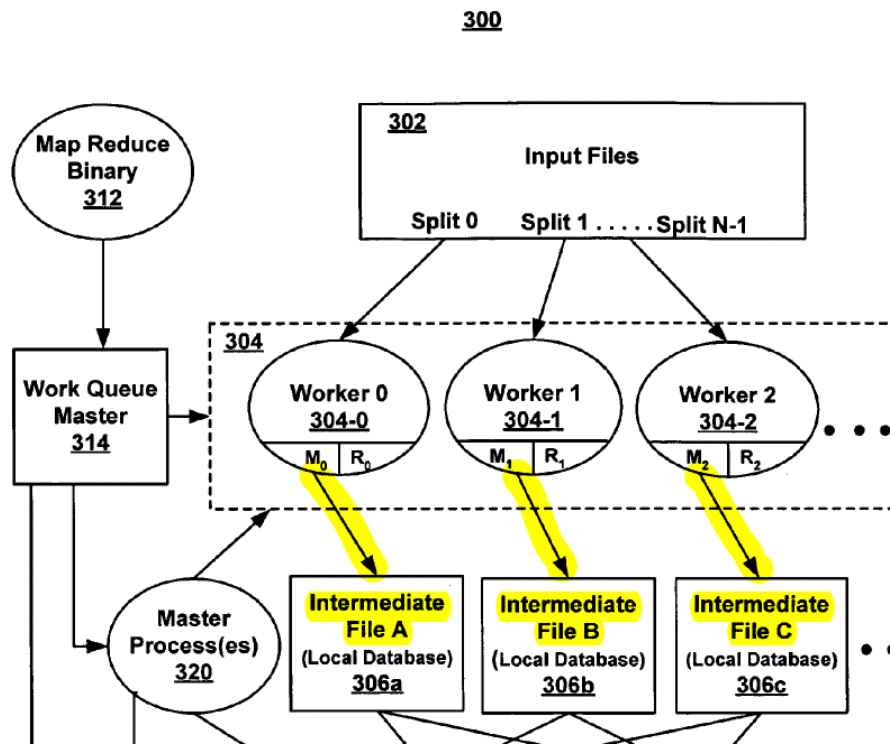
Combining Pike and Chowdhuri would have involved nothing more than combining two known techniques of scanning or mapping input data to produce intermediate results and reduce the same, with predictable results. EX-1005 ¶125. The combination would have been apparent to a POSITA because Pike and Chowdhuri disclose similar mapping and reducing operations, for similar purposes, suggesting their systems were compatible. *Id.* For example, Pike discloses "large-scale data processing in a parallel and distributed processing environment..." Pike, 1:20-26, 10:4-7, 2:17-41, 10:31-48. This is just like Chowdhuri, which discloses "process[ing] large data sets for both local and distributed systems," and "efficiently partition[ing] data so that multiple subtasks or operations can be performed simultaneously." Chowdhuri ¶¶11, 14. Chowdhuri also expressly discloses what

Pike suggests: performing these operations on data having different schema. *Id.* ¶¶127-135. A POSITA therefore, would have been motivated to make this combination to accommodate large collections of diverse data sets. EX-1005 ¶125.

The '610 patent also suggests this claim feature was conventional and available in MapReduce (including Pike's implementation): "[a]s discussed above with respect to the *conventional MapReduce architecture*, input, intermediate and output data sets *may each be characterized by their own schema*, and each schema operates according to 'key/value' pairs." '610 patent, 3:41-45. The patent also concedes MapReduce could be applied to input files of different schema, just allegedly not "efficiently." *Id.*, 3:15-20. Claim 1 does not claim mapping and reducing different input of schema more efficiently, it claims different schema as input, which is not patentable in light of the '610 patent's own admission.

[1g] "and the data of the first data group is mapped differently than..."

As shown in Figure 3 (below), Pike discloses or at least renders this feature obvious by disclosing that each of its data blocks (i.e., "split[s]") are provided to a different worker process 304-0, 304-1, and 304-2, which in turn each invoke a map operator, M₀, M₁, and M₂, respectively. Pike, 9:10-12, 9:58-10:27, 10:4-13 (each mapping a "respective input data block."); EX-1005 ¶¶127-28.



Pike, Fig. 3 (annotated).

Pike additionally discloses that its worker processes 304 output different lists of values for corresponding different intermediate data by disclosing that each map operator “generate[s] intermediate data values” that are “collected and written to one or more intermediate files 306.” *Id.*, 10:4-19; EX-1005 ¶128. Pike explains that, in certain embodiments, “each of the intermediate files 306 receives output from ***only one*** worker process 304, as shown in FIG. 3.” Pike, 10:4-19, 14:57-15:6, Fig. 3.

Pike’s disclosure of providing each data block (e.g., partition) to a different map operator, and outputting different intermediate data for each map operator, is consistent with the disclosure of the ’610 patent, which depicts these features in Figures 4 and 5. ’610 patent, Figs. 4, 5.

To the extent “mapped differently” is construed to require invoking different map functions, Pike still renders this feature obvious. EX-1005 ¶131. As described in Element [1f], Pike discloses that the map functions are application-specific and developed for specific use-cases. *See* V.A.[1f] (citing Pike, 2:22-26, 2:31-36, 3:59-64, 4:33-39, 6:1-12, 10:4-7, 10:39-50, 11:3638, 14:19-27). Pike’s system 400 also includes application software 422. Pike, 13:34-14:27, Fig. 4. Application software 422 includes “application-specific [map] functions for producing output.” *Id.* A POSITA would have understood that Pike contemplates multiple map functions in its application software, and that these map functions are invoked for processing particular data depending on the use-case, including using different map functions for a given map phase. EX-1005 ¶131. Indeed, Pike states this expressly by stating that “[g]enerally each ‘map process’ is a process configured (or configurable) to perform map *functions* and to execute an application-specific map operator.” Pike, 3:59-61.

[1h] “wherein the different schema and corresponding different intermediate data...”

Pike discloses a common key. Worker process 308 invokes a reduce function that “merges or otherwise combines the sorted intermediate data values *having the same key*, and writes the key and combined values to one or more output files 310.” Pike, 11:28-38; 14:65-15:3. To reduce on “the same key,” the input data must have “a key in common.” EX-1005 ¶¶132-33. Conventional MapReduce’s input data and

intermediate data also shared a common key. '610 patent, 1:17-27, 2:12-51 (common keys “k1 to k5”), Figs. 1, 2.

Because Pike discloses performing its operations on input data having different schema, Pike renders obvious different schema and different intermediate data having a common key (**Ground 1**). *See* Section V.A.[1f]; EX-1005 ¶¶134-35. Pike’s example of counting the occurrences of a word in a collection of documents also renders this feature obvious: the map operator “outputs a key/value pair for every word w in every document in the collection” and “[t]he Reduce function simply adds up the count values.” Pike, 10:31-11:4; EX-1005 ¶¶134-35.

Chowdhuri alternatively discloses (**Ground 2**) this by disclosing that its customer table 308 and order table 309 having a common key in the customer and order “state” (i.e., within the United States). Chowdhuri ¶¶126-127 (“group by state” and sorting “orders for each of the states”), ¶131 (“state codes”); EX-1005 ¶136. It would have been obvious to a POSITA that Pike’s mapping operators in the Pike-Chowdhuri system would map data blocks of customer table 308 and order table 309, and the resulting key-value pairs would have the customer and order “state” key in common. EX-1005 ¶136.

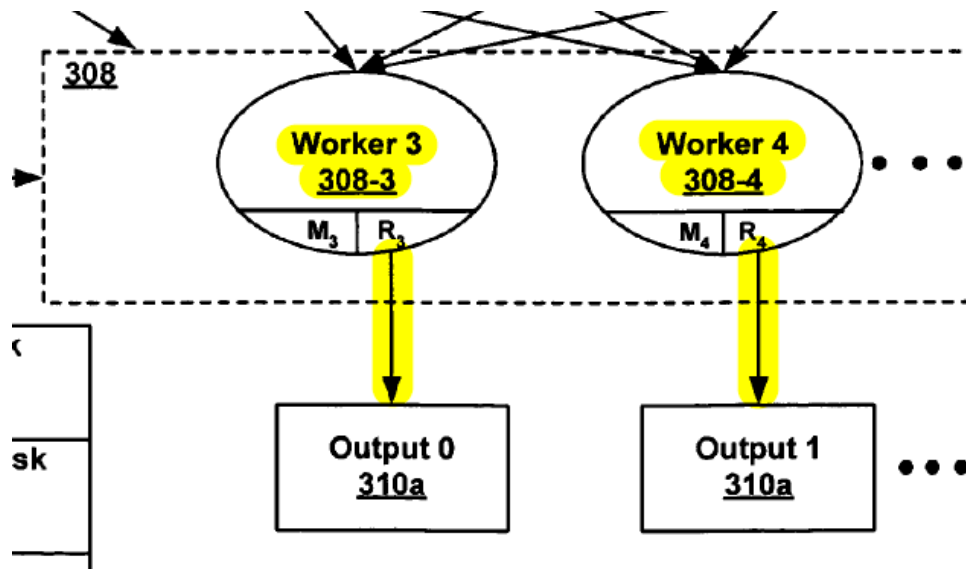
[1i] “reducing the intermediate data...”

Pike discloses this feature. As explained in Section V.A.[1f], the claim recites reducing intermediate data from “data groups” and is not limited to joining tables.

Pike discloses this by worker processes 308 invoking reduce operators to

reduce its intermediate data files 306. Pike, 11:6-55, Fig. 3. Each worker process 308 “*sorts* the intermediate data values in the subset of the intermediate files read by that worker process *in accordance with the key* of the key/value pairs in the intermediate data.” *Id.* Each worker process 308 further “*merges or otherwise combines the sorted intermediate data values having the same key*, and writes the key and combined values to one or more output files 310.” *Id.*; EX-1005 ¶137-38. Just like the ’610 patent, Pike discloses its worker processes 308 receive intermediate data from the intermediate files 306 and merge them into outputs.

Compare Pike, Fig. 3, with ’610 patent, Figs. 4, 5.



Pike, Fig. 3 (annotated).

Pike’s worker processes 308 also process intermediate data “for each data group in a manner that is defined to correspond to that data group” by tracking each reduce operation associated with an input file. *Id.*, 11:33-38, 13:5-17, 6:55-7:12;

see also Section V.A.[1e] (discussing worker process 320); EX-1005 ¶139. Pike “merg[es]” intermediate data based on a common key, as claimed, for example by adding the words found in a collection of documents. Pike, 10:31-11:4; *see* Section V.A.[1h]. Pike in view of Chowdhuri (**Ground 2**) also renders this feature obvious by disclosing joining, grouping, and sorting by customer and order “state.” *See* Section V.A.[1h].

To the extent claim 1 requires different reducing functions or reducing intermediate files differently, Chowdhuri discloses or at least renders this feature obvious, and a POSITA would have been motivated to implement this feature in Chowdhuri in order to reduce data according to the specificities of several input files, with a reasonable expectation of success. EX-1005 ¶141. Chowdhuri’s joining, grouping, and sorting iterators process intermediate data associated with customer and order tables, and join, group, and/or sort this intermediate data based on customer or order state. *Id.* This teaches reducing “differently” or employing different reduce functions—specific to each input file—just as the ’610 patent describes reducing intermediate data from an employee and department table. *Id.*; *Compare* Chowdhuri ¶¶126-27, 131, 135, *with* ’610 patent at Figs. 4, 5.

[1j] “wherein the mapping and reducing operations are performed by a distributed system.”

This feature was “conventional” in MapReduce. ’610 patent, 1:4-27, 3:9-15. Pike’s computer system 400 or data processing system 500 also discloses Pike’s

mapping and reducing operations being performed by a distributed system. *See* Section V.A.[1a]. Processing system 300 uses computer system 400, Pike, 13:53-14:27, 9:20-28, where “smaller tasks are spread across many machines,” *id.*, 13:37-50, 1:20-35; EX-1005 ¶142. “[D]istributed data processing system 500” also performs “the embodiments described [in Pike].” Pike, 2:43-49.

B. Claim 17

[17a] “A computer system...”⁷

Pike’s distributed data processing system 500 or computer system 400, carrying out the processes of system 300, discloses a computer system having computing devices with at least one processor and memory. *See* Section V.A.[1a], V.A.[1j]; Pike, 1:20-35; 2:43-3:35, 7:13-25, 11:57-12:33; EX-1005 ¶143. Pike’s computer system processes data of a data set that includes plurality of data groups in Pike’s input files, as discussed in Section V.A.[1b]. EX-1005 ¶143.

[17b] “partitioning...”

Pike discloses this feature. *See* Section V.A.[1c].

[17c] “providing...”

Pike discloses this feature. *See* Section V.A.[1d].

[17d] “to independently output...”

Pike discloses this feature. *See* Section V.A.[1e].

[17e] “wherein the data...”

⁷ *See* EX-1026 for a full claim listing.

Pike renders this feature obvious (**Ground 1**). *See* Section V.A.[1f]. Pike in view of Chowdhuri (**Ground 2**) also renders it obvious. *See* Section V.A.[1f].

[17f] “and the data of the first data group...”

Pike discloses this feature. *See* Section V.A.[1g].

[17g] “wherein the different schema...”

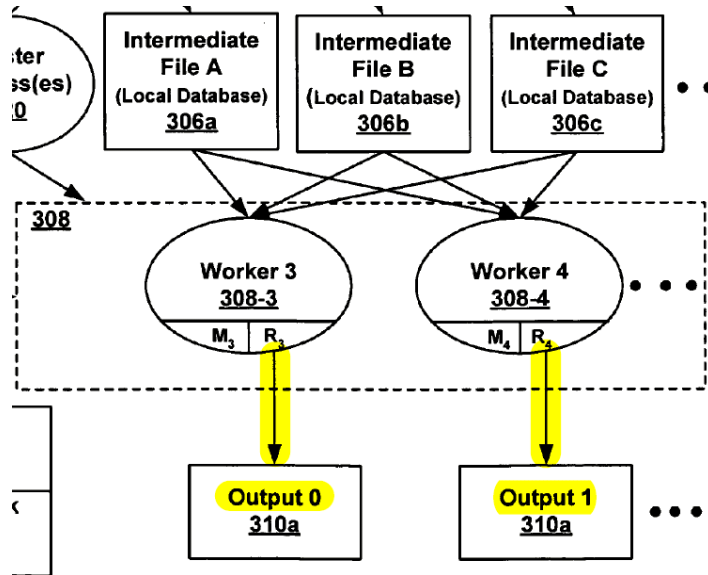
Pike and Chowdhuri both disclose this feature. *See* Section V.A.[1h].

[17h] “reduce...”

Pike discloses this feature. *See* Section V.A.[1i].

C. Claims 2 and 18 — “wherein: the at least one output data group is ...”

Claims 2 and 18 do not require the data groups to be the same as the “output data groups.” The claims only require reducing intermediate data to a plurality of “output data groups.” The specification confirms this. ’610 patent, 7:18-23. Pike discloses this feature by worker processes 308 reducing intermediate data 306 to “one or more output files 310.” Pike, 11:33-36, 11:6-55, 14:65-15:6, 6:65-7:12, Figs. 3, 7A; EX-1005 ¶151.



Pike, Fig. 3 (annotated).

D. Claims 3 and 19 — “wherein: corresponding intermediate data for a data group...”

The patent states that “the operations can collect information about the data that they are processing” and “save the information as metadata.” ’610 patent, 5:19-32, 7:31-35. One example is saving *keys* associated with a partition in the intermediate data. *Id.*, 5:22-32.

In the same manner, Pike discloses saving data associated with each partition’s key in its intermediate files 306. EX-1005 ¶¶152-53. Worker processes 308 invoke reduce operators to “retriev[e] intermediate *data values* from the intermediate data files 306” and sorts the same “in accordance with *the key*.” Pike, 11:8-14, 11:28-31. This enables worker process 308 to “merge[] or otherwise combine[] the sorted intermediate data values having the same key.” *Id.*, 11:33-36.

A POSITA would have understood Pike’s storing “values” in its intermediate

data files 306, and those values including a key “of the key/value pairs in the intermediate data,” discloses metadata associated with a manner of storage of the intermediate data, as claimed. EX-1005 ¶154. The key in the intermediate data is data about other data, which is encompassed by the plain meaning of “metadata.” *Id.*

Pike’s master process 320 tracking input files during its mapping and reducing phases, as well as associated task IDs, status, process IDs, and output files, also teaches “metadata . . . [that] is identifiable to the data group.” Pike, 6:55-7:25, Fig. 7A. Master process 320 associates the input files with output files (which include intermediate files). *See* Section V.A.[1e]. This teaches metadata “identifiable to the data group,” insofar as this limitation is portrayed in Patent Owner’s infringement contentions in its district-court case against Petitioner American Airlines. EX-1005 ¶155, EX-1012 and EX-1013.

Task Status Table					
Task ID	Status	Process	Input Files	Output Files	...
Map0000	Completed	P0000	12340	12341	...
Map0001	Completed	P0001	12344	12345	...
⋮	⋮				
Map0103	In Progress	P0010	12030	12031	...
Map0104	Failed	P0011	10101	12102	...
Map0103b	In Progress	P0010	12030	12031	...
⋮	⋮				
Red0000	In Progress	P0033	12340, ...	14000	...
Red0001	Waiting	-	-	-	...

Figure 7A

Pike, Fig. 7A (annotated).

To the extent argued otherwise, it would have been obvious to implement these features in Pike’s intermediate data files 306 as metadata, and a POSITA would have been motivated to do so to track those intermediate data files, with a reasonable expectation of success. EX-1005 ¶156. Doing so would have been nothing more than the simple substitution of one known element (i.e., metadata) for another (i.e., data values or status information in intermediate data) to obtain predictable results: storing this information as metadata, in the intermediate data files or otherwise. *Id.* A POSITA would have known that using identifiers or metadata to track data in a system like Pike was well known. *Id.* (citing EX-1024, EX-1025). Worker process 308 would access the keys from metadata to reduce the intermediate data to output data, or master process 320 would access status information to track and assign tasks. *Id.*

E. Claims 4 and 20 — “store[] [the corresponding intermediate data]...”

Pike discloses this feature. *See* Section V.A.[1e]. Pike additionally teaches this feature for the reasons it teaches “metadata.” *See* Section V.D; EX-1005 ¶157.

F. Claims 5 and 21 — “process[ing] the intermediate data... includes...”

Iterators are “useable within reduce functions *to access* intermediate data.” ’610 patent, 3:58-62. Pike’s worker process 308 (**Ground 1**) discloses an “iterator” that accesses data in intermediate data files 306 and applies a reduce operator to its

values. Pike, 11:6-38 (“retrieving intermediate data values” and sorting and merging them “in accordance with the key.”). Pike’s reducer operators are iterators because the inventors of Pike described them this way in the paper cited by the ’610 patent. EX-1009, 4 (“The reduce worker *iterates* over the sorted intermediate data . . . ”); EX-1005 ¶159. Pike worker processes 308 is an iterator that “corresponds to that data group” because it corresponds to the input file, as explained in Section V.A.[1i]. EX-1005 ¶¶158-59.

The patent’s description of iterators also corresponds with Chowdhuri’s, which are “self-contained software objects that accept a stream of rows from null or n-ary data sets.” Chowdhuri ¶61. The iterators “all open data streams, iteratively read the streams, process it and close the streams.” *Id.*; EX-1005 ¶¶158-60. Chowdhuri’s iterators disclose this feature (**Ground 2**) by employing a hash join iterator, GroupBy iterator, and sort iterator for reducing the results of a scanning operation. Chowdhuri ¶¶129-135, Figs. 3A, 3B; *see also* Section V.A.[1f]; EX-1005 ¶160.

It would have been obvious to a POSITA to combine Chowdhuri and Pike for the reasons explained in Section V.A.[1f]. A POSITA would have been motivated to implement Pike’s worker process 308 and reduce operators as an iterator because Pike suggests it, Chowdhuri discloses a mapping and reducing process using iterators, and for the reasons explained in Chowdhuri: “iterators... can be easily duplicated for horizontal parallelism across CPU and I/O threads.” Chowdhuri ¶131; EX-1005 ¶161. Implementing Pike’s worker process 308 as an iterator would

promote parallelism in system 300 and reduce run times. EX-1005 ¶161.

G. Claims 6 and 23

[a] “wherein: the intermediate data includes...”

Pike’s intermediate data files 306 including key/value pairs disclose this feature. Pike, 11:27-31, 7:34-52, 10:31-58. It would have been obvious to a POSITA that an intermediate data file 306 that includes key/value pairs is a “grouped set[] of key value pairs” because it is a group of data that includes key/value pairs. EX-1005 ¶162.

Pike’s “one or more worker processes 304 and 308” also disclose a reducing step carried out by a plurality of reducers. Pike, 8:65-28, 11:6-55, 13:5-17, 14:65-15:6, Fig. 3 (items 308-3, 308-4); EX-1005 ¶163.

[b] “partition[ing] the intermediate data...”

Pike discloses this by its partition operator “specify[ing] that all intermediate data is to be directed to an intermediate file corresponding to the value of the first byte of the key.” Pike, 7:40-57, 14:9-27, 14:63-65, Fig. 4. It would have been obvious to a POSITA that Pike’s partition operator partitions intermediate data into a plurality of partitions, each having separate key/value pairs. EX-1005 ¶164.

Pike further discloses that “[i]n some embodiments, each of the worker processes 308 reads a corresponding one of the intermediate files 306 produced by all or a subset of the worker processes 304.” Pike, 11:14-27. A POSITA would have understood that this teaches providing the intermediate data of each partition to a

separate one of the reducers, as claimed. EX-1005 ¶165.

Pike at least renders this feature obvious. *Id.* ¶166. Pike discloses that the partition operator is application-specific and determined by the user and use-case. Pike, 14:12-27, 4:49-53. A POSITA would have found it obvious to implement Pike so that each intermediate file includes only one group of key-value pairs, and each partition is provided to a different reducer. This would have been, to a POSITA, a simple matter of partition design choice rendered obvious by Pike. EX-1005 ¶166. The '610 patent also admits this feature was conventional. '610 patent, 4:63-67 (“similar to what is done by the conventional MapReduce”), 2:61-64 (partition functions 206a, 206b, 206c), Fig. 2.

H. Claims 7 and 24

[a] “the intermediate data includes...”

Pike discloses this feature. *See* Section V.G.[a].

[b] “partition[ing] the intermediate data...”

Claims 7 and 24 do not require “some of the key/value pairs of the intermediate data” to *each* be provided to “more than one of the partitions” (i.e. providing the same key/value pair to more than one partition). The claims recite only providing at least some of the key/value pairs to more than one partition (e.g., providing two key/value pairs to two partitions).

Pike’s partition operator discloses partitioning the intermediate data into a plurality of partitions and providing each partition to a reducer. *See* Section V.G.[b];

EX-1005 ¶168. Pike also discloses providing some of the key/value pairs to more than one partition by portioning by “a modulo function,” where the application programmer “specifies the modulus.” Pike, 7:52-55. A POSITA would have understood that dividing the intermediate data based on a modulus, rather than dividing them by key, teaches providing key/value pairs of the intermediate data to multiple partitions. EX-1005 ¶169. This at least suggests providing some of the key/value pairs to multiple partitions and therefore renders this feature obvious. *Id.*

To the extent claims 7 and 24 require providing the same key/value pair to multiple partitions, this would have been an obvious implementation of MapReduce in Pike (**Grounds 1 and 2**). To make a particular key-value pair go to more than one partition, the map operation would simply emit the same key/value pair multiple times. EX-1005 ¶170. Repeating the same process twice is not new and would yield expected results: emitting the same key/value pair more than once, to provide each to a separate partition for reduction. *Id.* This is a straightforward application of a map function in “conventional MapReduce” that a POSITA would have found obvious to implement. A POSITA would have been motivated to do so to achieve a desired design choice for the partitions disclosed in Pike, with a reasonable expectation of success. *Id.*

This is also obvious based on Chowdhuri (**Ground 2**), which discloses the use of exchange operators 371, 372. Chowdhuri ¶134-35, Fig. 3B; EX-1005 ¶171. The exchange operators “take[] the data from the producer tasks and is capable of

redistributing data for a different degree of parallelism in its consuming tasks.”

Chowdhuri ¶134. The redistribution “is extended to redistribution of data to *multiple servers* on a cluster for processing the data.” *Id.*

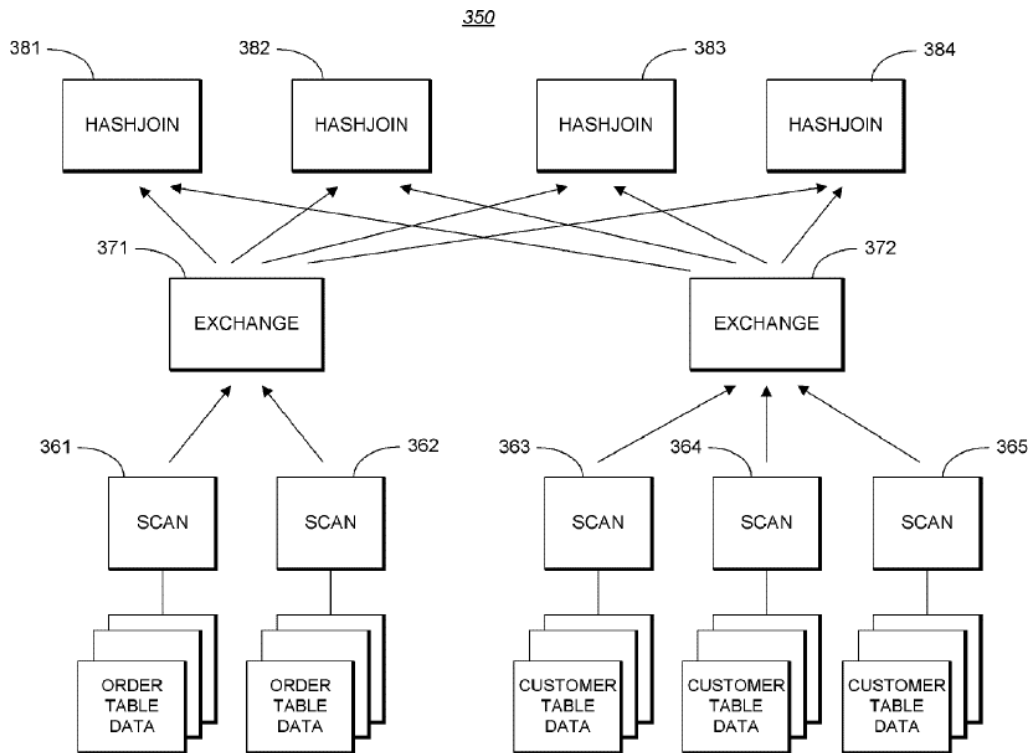


FIG. 3B

Id., Fig. 3B.

Chowdhuri also discloses, in a “replicated join,” that “each tuple from a given partition on the outer operand will be sent to *every partition* of the inner operand, which means that if the same tuple encounters a match in more than one partition.” *Id.* ¶227. In addition to understanding that it was known to send values, including key/value pairs, to more than one partition for processing, a POSITA would have been motivated to do so in Pike based on Chowdhuri. EX-1005 ¶172. Implementing these teachings

into Pike's system 300 would have been nothing more than the simple substitution of known features for predictable results: providing all or some of Pike's key/values to more than one partition. *Id.*

I. Claims 8 and 25 — “wherein: in the partitioning step all of the key/value pairs of the intermediate data are provided to all of the partitions.”

Claims 8 and 25 do not require all of the key-value pairs to be provided to *each* of the partitions. Accordingly, Pike teaches this feature by disclosing that all of its key/value pairs are provided to its partitions. *See* Section V.G.[b].

To the extent claims 8 and 25 require each key/value pair to be provided to each partition, Pike renders this feature obvious (**Grounds 1 and 2**). A POSITA would have understood that broadcasting each key/value pair to each partition would be one of several options for partitioning a set of keys. EX-1005 ¶174. As explained for claims 6 and 23, the partition operators of Pike are application-specific and will vary with use-case. *See* Section V.G.[b]. Given the available option to provide each key/value pair to separate partitions, it would have been obvious to provide each key/value pair to each partition. EX-1005 ¶174. A POSITA would have been motivated to do so to increase fault tolerance. *Id.* (citing Pike, 4:30-39, 6:37-7:12, 9:20-28, 11:54-55).

J. Claims 9 and 26

[a] “wherein: at least some of the reducers include...”

Pike in view of the conventional art admitted by the '610 patent discloses this

feature (**Ground 1**). Pike discloses a sort and combine task by worker process 308 “sort[ing] the intermediate data values” by key, Pike, 11:28-38, 12:7-17, 14:65-15:6, 9:45-49, and “merg[ing] or otherwise combin[ing] the sorted intermediate data values having the same key,” *id.*, 11:28-38. While Pike does not expressly disclose that its worker processes 308 include a “group-by-key,” the ’610 patent admits this feature was “conventional” in MapReduce. ’610 patent, 2:9-67, 2:64-67, Fig. 2 (item 210a, 210b). A POSITA would have known that Pike’s worker process 308 includes this MapReduce feature, or a POSITA would have known that it was conventional and implemented it in Pike. EX-1005 ¶¶175-76. Doing so would have been obvious and the implementation of a known MapReduce feature in an existing MapReduce system in system 300. *Id.* A POSITA would have been motivated to do so to make the processing of the output data “more efficient and/or convenient,” as the ’610 patent concedes. *Id.*; ’610 patent, 2:64-67.

Pike in view of Chowdhuri (**Ground 2**) also renders this feature obvious. As explained in claim 1, Chowdhuri discloses reducing data using a hash join iterator 303, GroupBy iterator 304, and sort iterator 305. *See* Section V.A.[1f]; Chowdhuri, ¶¶127, 130. A POSITA would have been motivated to combine Pike and Chowdhuri to implement Pike’s worker processes 308 as Chowdhuri’s iterators for the reasons explained in Section V.A.[1f]. EX-1005 ¶177.

[b] “generating and providing metadata for...”

Pike’s storing “values” in its intermediate data files 306, including a key, and

the status of the mapping process, teaches providing metadata during at least the mapping phase. *See* Section V.D. (citing Pike, 11:28-31, 6:55-7:12, Fig. 7A).

Pike also teaches providing metadata during the sorting and combining by disclosing that “[e]ach worker process 308 *sorts* the intermediate data *values* in the subset of the intermediate files,” Pike, 11:28-31, and “merg[es] or otherwise combin[es] the sorted intermediate data *values*” by key, *id.*, 11:33-38. EX-1005 ¶¶178-79.

Pike also teaches this feature by its master process 320 tracking associated input files during its mapping and reducing phases, as explained in Section V.D. *See also* Pike, 6:55-7:25, Fig. 7A; EX-1005 ¶180. Master process 320 associates the input files (e.g., “data groups”) with output files, including intermediate files. *See* Section V.A.[1e].

As explained with respect to claims 3 and 19, Pike discloses or renders obvious storing this information as metadata. *See* Section V.D; EX-1005 ¶181. Pike therefore discloses or renders claims 9 and 26 obvious.

K. Claims 10 and 27 — “wherein: the reducing [step] includes processing the metadata.”

Pike discloses this feature by sorting the values in the intermediate data files during the reducing. As explained in claims 3, 9, 19, and 26, Pike’s storing “values” in its intermediate data files 306, including a key “of the key/value pairs in the intermediate data,” teaches generating and providing metadata during the mapping

and reducing. *See* Section V.D, V.J.[b]; EX-1005 ¶182. During reducing, the worker process 308 “sorts the intermediate data values in the subset of the intermediate files read by that worker process in accordance with the key of the key/value pairs in the intermediate data.” Pike, 11:28-31. Pike’s merging the sorted intermediate data values by key also discloses this metadata processing. *Id.*, 11:33-38; EX-1005 ¶182. Finally, Pike’s master process 320 processing status data when reducing is complete also teaches processing the metadata. Pike, 11:49-55; EX-1005 ¶182; *see also* Sections V.D. (citing Pike, 6:55-7:12, Fig. 7A), V.J.[b].

L. Claims 11 and 28 — “wherein: ...process[ing] the intermediate data for each data group in a manner that is defined to correspond to that data group includes...”

For the reasons explained in claims 5 and 21, Pike’s worker process 308 (**Ground 1**), and Pike’s worker process 308 in view of Chowdhuri’s iterators (**Ground 2**) disclose processing the intermediate data for each group in a manner that is defined to correspond to that data group by employing an iterator for each data group that corresponds to that data group. *See* Section V.F; EX-1005 ¶183.

As explained in claims 9, 10, 26, and 27, Pike further discloses providing metadata to the processing of the reducing step by providing values of the intermediate data to the worker process 308 for sorting and merging, and providing status information associated with the reducing to master process 320. *See* Section V.J.[b], V.K; EX-1005 ¶184.

M. Claims 12 and 22

- [a] **“wherein: the intermediate data includes a plurality of grouped sets of key/value pairs” and reducing step via “a plurality of reducers;”**

Pike discloses this for the reasons explained in claim 6. *See* Section V.G.[a].

- [b] **“for at least one of the reducers, the iterator corresponding to a particular data group, for that reducer, operates...”**

Pike’s worker process 308 (**Ground 1**), and Pike’s worker process 308 in view of Chowdhuri’s iterators (**Ground 2**), disclose processing the intermediate data for each group in a manner that is defined to correspond to that data group by employing an iterator for each data group. *See* Section V.F. Pike renders obvious and Chowdhuri discloses that the data groups (i.e., input files) may have different schema. *See* Section V.A.[1f]. From this, a POSITA would have understood that Pike alone and in view of Chowdhuri also renders obvious using two iterators to sort different keys from intermediate data associated with two input files having different schema. EX-1005 ¶186. This satisfies claims 12 and 22.

Pike discloses that its worker processes 308 “sort[] the intermediate data values in the subset of the intermediate files read by that worker process in accordance with the key of the key/value pairs in the intermediate data,” and that “sorting of the key/value pairs is an application independent function of the reduce threads in the worker processes 308.” Pike, 11:28-33. A POSITA would understand therefore, that Pike’s sorting is performed on multiple keys across the intermediate data. EX-1005 ¶187. Sorting multiple keys in MapReduce was also conventional,

as the '610 patent concedes in its “Sort and Group” tasks 210a, 201b. '610 patent, 2:9-67, 2:64-67, Fig. 2.

A POSITA would have found it obvious to implement Pike’s sort process, or Chowdhuri’s sort iterator 305, to sort different keys corresponding to different data groups. EX-1005 ¶188. Pike suggests sorting different keys and operating on different input data, like Chowdhuri. *Id.* And the '610 patent admits sorting different keys among different partitions was known. '610 patent, 2:9-51 and Fig. 1. Implementing Pike’s or Chowdhuri’s sorting to sort different keys associated with different “data groups,” at most would have been a straightforward modification based on the suggestions of Pike and the known features of MapReduce, which would have been well-within the capabilities of a POSITA. *Id.*

N. Claims 13 and 29 — “wherein: the intermediate data processing step of the reducing step further comprises...”

The '610 patent states that data that is not intermediate data “can come from data local to [a] reduce function or data shared by other reduce functions.” '610 patent, 5:66-6:7. A POSITA would have understood that this data includes data received from sources other than map functions, and includes data provided by other reduce functions. EX-1005 ¶189.

Pike discloses this feature. Pike’s worker processes 308 receiving information from master process 320 that “informs the worker processes 308 *where* to find intermediate data files 306 and schedul[ing] read requests for retrieving intermediate

data values from the intermediate data files 306,” teaches processing data that is not intermediate data. Pike, 11:8-27. The data is received from the master process 320 and not from a map function, and therefore is not intermediate data. EX-1005 ¶190. It is also processed by worker process 308 to locate intermediate data to reduce. *Id.*

Chowdhuri’s GroupBy and sort iterators 304, 305 also teach processing data that is not intermediate data because these receive data from another reducing function: the hash join iterator 303. Chowdhuri ¶¶127-135, Fig. 3A. As explained in claims 1 and 17, a POSITA would have understood that Chowdhuri’s GroupBy and sort iterators 304, 305 are reducing functions. *See* Section V.A.[1f], V.B.[17e]. A POSITA moreover would have understood these iterators receive data from another hash join reducing function, and that the received data is not intermediate data. EX-1005 ¶191.

O. Claims 14 and 30 — “...a plurality of reducers...” and “the data that is not intermediate data, for a particular reducer, includes...”

Pike (**Ground 1**), and Pike in view of Chowdhuri (**Ground 2**), disclose this feature. EX-1005 ¶192. Pike discloses a plurality of reducers for the reasons explained in claim 6, 13, 23, and 29. *See* Section V.G.[a], V.N. Chowdhuri’s GroupBy and sort iterators also teach this. *See* Section V.N. For the reasons explained in claims 13 and 29, both Pike and Chowdhuri disclose processing data that is not intermediate data that is associated with another reducer. *See id.*

P. Claims 15 and 31 — wherein reducing is carried out via “a plurality of reducers,” and “the data that is not intermediate data,

for a particular reducer, includes data that is associated with that reducer.”

Pike discloses a plurality of reducers for the reasons explained in claims 6 and 23. *See* Section V.G.[a]. Pike also discloses processing data that is not intermediate data by a particular reducer and associated with that reducer, as explained in claims 13 and 29, by Pike’s worker processes 308 receiving information from master process 320 that “informs the worker processes 308 where to find intermediate data files 306 and schedul[ing] read requests for retrieving intermediate data values from the intermediate data files 306.” *See* Section V.N. (citing Pike, 11:8-27). The data is received from the master process 320 and not from a map function, and therefore is not intermediate data. EX-1005, ¶193. It is also processed by worker process 308 that invokes a reduce function, and is data associated with that reduce function (e.g., where to locate intermediate data to reduce). *Id.*

Q. Claims 16 and 32 — reducing by “relating the data among the plurality of data groups.”

Pike discloses this feature by its worker processes 308 invoking reducing operators to “merge[] or otherwise combine[]” the intermediate data files 306. Pike, 4:40-44, 11:33-41. A POSITA would have understood that merging or combining intermediate data across Pike’s input files (e.g., Pike’s “data groups”) teaches “relating the data among the plurality of data groups.” EX-1005 ¶194. This is consistent with the written description of the ’610 patent, which discloses “merg[ing] all intermediate data over all groups,” including data from an employee table and

department table. '610 patent, 4:50-59.

R. Claim 33

[33a] “A map-reduce...”

Pike’s large-scale data processing system 300 discloses a “map-reduce method” because it maps and reduces data. Pike, 2:12-39, 2:50-3:35, 6:557:25, 8:65-11:55; EX-1005 ¶195. System 300 processes data from a plurality of groups over a computer system for the reasons explained in Sections V.A.[1a], V.A.[1j], and V.A.[1b]. Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) renders obvious data groups of “different schema” for the reasons explained in Section V.A.[1f].

[33b] “for a first data set...”

For the reasons explained in Sections V.A.[1b], V.A.[1f], Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious. A POSITA would have understood that both Pike and Chowdhuri disclose a “data set” with key-value pairs “belong[ing] to a . . . data group.” EX-1005 ¶196. Pike discloses input files, as discussed in Section V.A.[1b]. Chowdhuri discloses input tables, as discussed in Section V.A.[1f] (claim 1, element [1f]). Pike also suggests and renders obvious input files of different schema, and Chowdhuri discloses this expressly. *See* Sections V.A.[1b], V.A.[1f]; EX-1005 ¶196.

[33c] “partitioning...”

Pike’s splitting input files into data blocks discloses this feature for the reasons

explained in Section V.A.[1c]. It was also conventional. *See id.*

[33d] “providing...”

Pike’s providing each data block to a worker processes 304 to invoke mapping operators discloses this feature for the reasons explained in Section V.A.[1d]. It was also conventional. *See id.*

[33e] “to independently output...”

Pike’s worker processes 304 outputting intermediate files 306 discloses this feature for the reasons explained in Section V.A.[1e]. A POSITA would have understood that the “resulting key-value pairs” of mapping in the Pike would include values taken from Pike’s input files (**Ground 1**), or from Chowdhuri’s order table 308 (**Ground 2**). EX-1005 ¶199.

[33f] “for a second data set...”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Sections V.A.[1b], V.A.[1f], V.R.[33b].

[33g] “partitioning...”

This element recites partitioning the *first* data set into a plurality of data partitions. Whether this is understood to be a second partitioning of the first data set, or a single partition of the first data set into a plurality of data partitions, Pike’s splitting its input files (**Ground 1**) or splitting Chowdhuri’s customer table 308 (**Ground 2**) into data blocks renders this feature obvious, as explained in Sections V.R.[33c], V.A.[1c]; EX-1005 ¶201.

For example, it would have been obvious to split a portion of Pike's input files or Chowdhuri's customer table 308 into a first set of data blocks, and provide those data blocks to a first set of mapping operators to form first intermediate data, and then split another portion of Pike's input files or Chowdhuri's customer table 308 to provide data blocks to a second set of mapping operators to form second intermediate data. EX-1005 ¶202. Pike suggests this by disclosing splitting an input into N-1 data blocks that can each be provided to different mapping functions. Pike, 9:65-10:15, Fig. 3. A POSITA would have been motivated to implement this feature by modifying Pike and/or Chowdhuri to achieve a desired distribution of partitions among the computing resources that run the mapping operators, with a reasonable expectation of success. EX-1005 ¶202.

To the extent it is argued that this feature requires partitioning the second data set, Pike's splitting of its input files, or splitting Chowdhuri's order table 309 into data blocks in the Pike-Chowdhuri system, renders this feature obvious as explained in Sections V.R.[33c], V.R.[33f]; V.A.[1c]. EX-1005 ¶203.

[33h] “providing...”

Forming “differing” sets of resulting key value pairs is satisfied by mapping functions that each output, for example, a different number of key value pairs. Finding a particular word once in one partition, for example, and two times in another partition, satisfies this limitation Pike discloses or at least suggests this by applying an “application-specific map() operator to [a] respective input data block

so as generate intermediate data values,” mapping “every word w in every document in the collection, where the key/value pair is $\langle w, 1 \rangle$.” Pike, 9:56-10:15, 10:31-43. A POSITA would have understood that Pike’s mapping operators would output particular numbers of the same key for different data blocks, and that the number output from each map operator would differ from block to block. EX-1005 ¶ 204.

Even if this element requires mapping using different keys, Pike discloses or renders this obvious by its mapping operators mapping different words in a document. Pike, 9:56-10:15, 10:31-43. Mapping different words teaches or at least suggests forming first and second intermediate data having differing first and second key-value pairs, where the different words are the different keys. Pike’s outputting a plurality of intermediate files 304 also teaches forming first and second intermediate data sets. *Id.*, 9:65-10:19, Fig. 3; EX-1005 ¶205.

[33i] “wherein the first schema differs from the second schema,”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Sections V.R.[33a], V.A.[1f].

[33j] “wherein the first and second schema...”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.A.[1h]. For example, Pike’s mapping operators forming resulting key-value pairs based on a single word teaches this feature. *See* Section V.R.[33h].

193. Chowdhuri alternatively discloses this by disclosing a customer and

order table 308, 309 having a common key: customer and order “state.” Chowdhuri ¶¶126-127, ¶131; *see* Sections V.A.[1f], V.A.[1h]. Pike’s mapping operators in the Pike-Chowdhuri system would have mapped data blocks of the customer table 308 (or data blocks in both customer and order tables 308, 309), and the resulting key-value pairs would have a “state” in common. EX-1005 ¶¶207-08.

[33k] “reducing...”

Pike discloses reducing intermediate data sets together to form output data for the reasons explained in Section V.A.[1i]. Pike and Chowdhuri also disclose reducing by iterating on a key from the intermediate data for the reasons explained in Section V.F. *See also* Pike, 11:6-38; Chowdhuri ¶¶129-135; EX-1005 ¶209. In the Pike-Chowdhuri system, a POSITA would have understood that iterating by a specific word or by customer state teaches iterating on at least one key from the first and second intermediate data set. *See* Section V.R.[33h]; EX-1005 ¶209.

[33l] “the output data set...”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious. For example, a POSITA would have understood that Pike’s word count in a collection of documents (i.e., the output of its reducing) would have a different format, or “schema,” than the input documents. EX-1005 ¶210.

A POSITA at least would have found this obvious based on a POSITA’s knowledge of MapReduce, which describes counting words in documents that are webpages scraped from the Internet. EX-1009, 1-3; EX-1005 ¶210. A POSITA

would have recognized that a word count output from Pike’s reducing operation would have a different schema than an input website. EX-1005 ¶210.

The Pike-Chowdhuri system further renders this feature obvious by outputting a merged list of customers and orders based on a common state. *See* Section V.R.[33h]. Chowdhuri’s merged list of customers and orders based on state suggests an output with a different schema than the input customer and order tables, which would likely contain other fields than just customer names, orders, and states. EX-1005 ¶211. This is consistent with the ’610 patent’s merging tables into a combined department-employee table. ’610 patent, 3:23-34, Figs. 4, 5.

[33m] “wherein the mapping and reducing operations...”

Pike discloses this for the reasons explained in Section V.A.[1j].

S. Claim 40

[40a] “A computer system...”

Pike discloses a computer system with a plurality of computing devices with at least one processor and memory for the reasons explained in Section V.B.[17a]. Pike discloses processing data of a data set that comprises a plurality of data groups for the reasons explained in Sections V.A.[1a], V.A.[1b].

[40b] “for a first data set having a plurality of first key-value pairs”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.R.[33b].

[40c] “partitioning the first data set...”

Pike discloses this feature for the reasons explained in Section V.R.[33c].

[40d] “providing each data partition...”

Pike discloses this feature for the reasons explained in Section VI.A.[33d].

[40e] “to independently output...”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.R.[33e].

[40f] “for a second data set”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.R.[33f].

[40g] “partitioning the first data set”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.R.[33g].

[40h] “providing each data partition...”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.R.[33h].

[40i] “wherein the first schema...”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.R.[33i].

[40j] “reducing the first and the second intermediate data set together...”

Pike (**Ground 1**) and Chowdhuri (**Ground 2**) disclose this feature for the reasons explained in Section V.R.[33k].

[40k] “and the output data set...”

Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Section V.R.[33l].

[40l] “wherein the first and second schema and the first and second set of resulting key-value pairs...”

Pike discloses this feature for the reasons explained in Section V.R.[33m].

T. Claims 34 and 41

Reducing the intermediate data based on the common key of the intermediate data to merge the first and second intermediate data satisfies this limitation. Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) render this feature obvious for the reasons explained in Sections V.R.[33j], V.R.[33k]. EX-1005 ¶225.

U. Claims 35 and 42

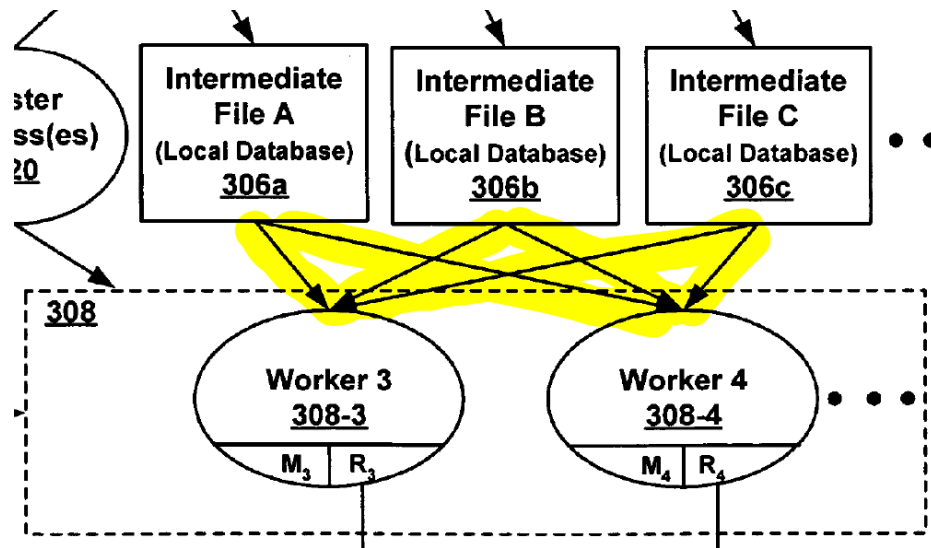
Pike (**Ground 1**) and Pike in view of Chowdhuri (**Ground 2**) disclose this for the reasons explained in claims 7 and 24. *See* Section V.H.

V. Claims 36 and 43

These claims do not recite providing all key-value pairs to *each* reducer, only that the key-value pairs are all provided to reducers. Accordingly, Pike teaches this feature by disclosing that all of its key/value pairs are provided to its partitions, which are reduced by worker processes 308. *See* Sections V.G.[b], V.I.

Even if all key-values must be provided to each partition (and thus each reducer), Pike renders this feature obvious for the reasons explained in Section V.I. Pike discloses that data from each intermediate data file 306 is provided to all of its

worker processes 308, as reflected below in Figure 3. Pike, 11:6-55, Fig. 3. Pike also suggests that its worker processes 308 read more than one intermediate data file, and even all intermediate files. *Id.*; EX-1005 ¶¶227-228.



Pike, Fig. 3 (annotated).

Pike also depicts the data from its intermediate data files 306 being distributed to each worker process 308. *Id.* This renders obvious providing each key-value pair in the input data to all reducers. EX-1005 ¶229. A POSITA would have implemented Pike to provide all key-value pairs to the reducers to account for all key-value pairs in the input data sets. *Id.* Indeed, Pike suggests this by disclosing its reduce tasks may provide statistics about the key-value pairs in the input data sets. Pike, 11:38-55. The most accurate statistics account for all available data in the data sets, including all of the key-value pairs. EX-1005 ¶229.

W. Claims 37 and 44

Pike and Chowdhuri render this feature obvious for the reasons explained in

Section V.J.

X. Claims 38 and 45

Pike and Chowdhuri render this feature obvious for the reasons explained in Section V.K.

Y. Claims 39 and 46

Pike and Chowdhuri render this feature obvious for the reasons explained in Section V.L.

VI. Mandatory Notices Under 37 C.F.R. § 42.8

A. Real Party-in-Interest

The real parties-in-interest for this Petition are American Airlines, Inc. and Hilton Domestic Operating Company, Inc.

B. Related Matters

To the best knowledge of Petitioners, the '610 patent is or has been involved in the following judicial matters:

Name	No.	Court	Filed
<i>R2 Solutions LLC v. Hilton Domestic Operating Company Inc.</i>	3-22-cv-02761	N.D. Tex.	December, 21, 2022
<i>FedEx Corporate Services, Inc. v. R2 Solutions LLC</i>	IPR2022-01456	PTAB	August, 25, 2022
<i>In re: FedEx Corp. Serv.'s, Inc.</i>	22-156	Fed. Cir.	July 21, 2022

<i>Allegiant Travel Company v. R2 Solutions LLC</i>	2-22-cv-00828	D. Nev.	May. 24, 2022
<i>R2 Solutions LLC v. American Airlines, Inc.</i>	4-22-cv-00353	E.D. Tex.	Apr. 28, 2022
<i>R2 Solutions LLC v. CVS Health Corporation et al</i>	4-22-cv-00354	E.D. Tex.	Apr. 28, 2022
<i>R2 Solutions LLC v. Hilton Worldwide Holdings Inc. et al</i>	4-22-cv-00356	E.D. Tex.	Apr. 28, 2022
<i>R2 Solutions LLC v. Citigroup Inc.</i>	4-22-cv-00357	E.D. Tex.	Apr. 28, 2022
<i>R2 Solutions LLC v. FedEx Corporate Services, Inc.</i>	4-21-cv-00940	E.D. Tex.	Nov. 29, 2021
<i>R2 Solutions LLC v. State Farm Mutual Automobile Insurance Company</i>	4-21-cv-00941	E.D. Tex.	Nov. 29, 2021
<i>R2 Solutions LLC v. Booking.com BV</i>	4-21-cv-00942	E.D. Tex.	Nov. 29, 2021
<i>R2 Solutions LLC v. Booking.com Transport Limited</i>	4-21-cv-00943	E.D. Tex.	Nov. 29, 2021
<i>R2 Solutions LLC v. Agoda Company Pte. Ltd.</i>	4-21-cv-00944	E.D. Tex.	Nov. 29, 2021
<i>R2 Solutions LLC v. Expedia Group, Inc.</i>	6-21-cv-00628	W.D. Tex.	Jun. 17, 2021
<i>R2 Solutions LLC v. iHeartMedia, Inc.</i>	6-21-cv-00552	W.D. Tex.	Jun. 01, 2021
<i>R2 Solutions LLC v. JPMorgan Chase & Co.</i>	4-21-cv-00174	E.D. Tex.	Mar. 02, 2021
<i>R2 Solutions LLC v. The Charles Schwab Corporation</i>	4-21-cv-00122	E.D. Tex.	Feb. 09, 2021
<i>R2 Solutions LLC v. Fidelity Brokerage Services LLC</i>	4-21-cv-00123	E.D. Tex.	Feb. 09, 2021
<i>R2 Solutions LLC v. Deezer SA</i>	4-21-cv-00090	E.D. Tex.	Jan. 29, 2021

<i>R2 Solutions LLC v. Walmart Inc.</i>	4-21-cv-00091	E.D. Tex.	Jan. 29, 2021
<i>R2 Solutions LLC v. Target Corporation</i>	4-21-cv-00092	E.D. Tex.	Jan. 29, 2021
<i>R2 Solutions LLC v. Workday, Inc.</i>	4-21-cv-00093	E.D. Tex.	Jan. 29, 2021

C. Lead and Backup Counsel

Lead Counsel	Back-Up Counsel
John B. Campbell Reg. No. 54,665 jcampbell@McKoolSmith.com McKool Smith, P.C. 303 Colorado, Suite 2100 Austin, Texas 78701 Telephone: (512) 692-8700 Facsimile: (512) 692-8744	Geoffrey Smith (to be admitted pro hac vice) gsmith@McKoolSmith.com McKool Smith, P.C. 303 Colorado, Suite 2100 Austin, Texas 78701 Telephone: (512) 692-8700 Facsimile: (512) 692-8744 Raphael Chabaneix Reg. No. 78,922 rchabaneix@McKoolSmith.com McKool Smith, P.C. 300 Crescent Court, Suite 1500 Dallas, Texas 75201 Telephone: (214) 978-4000 Facsimile: (214) 978-4044 Steven Peters Reg. No. 73,193 speters@McKoolSmith.com McKool Smith, P.C. 1999 K Street, NW Suite 600 Washington, DC 20006 Telephone: (202) 370-8300 Facsimile: (202) 370-8344

	David E. Finkelson (to be admitted pro hac vice) dfinkelson@mcguirewoods.com McGuireWoods LLP 800 East Canal Street Richmond, VA 23219-3916 Telephone: (804) 775-1000 Facsimile: (804) 698-2016 Kyle Smith Reg. No. 78,579 ksmith@mcguirewoods.com McGuireWoods LLP 501 Fayetteville St. Raleigh, NC 27601 Telephone: (919) 755-6600 Facsimile: (919) 755-6699
--	--

D. Service Information

Please address all correspondence to lead and backup counsel at the addresses shown above. Petitioners also consent to electronic service by e-mail.

VII. Grounds for Standing

Petitioners certify that the '610 patent is available for *inter partes* review, and that Petitioners are not barred or estopped from requesting *inter partes* review to challenge the claims on the grounds identified herein. Petitioners filed this Petition within one year of service of the original complaints against Petitioners in the district court litigation.

VIII. Conclusion

Petitioners have established a reasonable likelihood of prevailing with respect to the challenged claims and requests institution of *inter partes* review and cancellation of each challenged claim as unpatentable.

Date: March 6, 2023

Respectfully submitted,

/s/ John Campbell

John B. Campbell, Lead Counsel
Reg. No. 54,665
Telephone: (512) 692-8700
Facsimile: (512) 692-8744
jcampbell@McKoolSmith.com
McKOOL SMITH, P.C.
303 Colorado, Suite 2100
Austin, Texas 78701
Counsel for Petitioners

CERTIFICATE OF WORD COUNT

The undersigned hereby certifies that the foregoing Petition for *Inter Partes* Review contains 13,994 words, excluding those portions identified in 37 C.F.R. § 42.24(a), as measured by the word-processing system used to prepare this paper.

Date: March 6, 2023

/s/ John Campbell
John Campbell (Reg. No. 54,665)

CERTIFICATE OF SERVICE

The undersigned certifies that, in accordance with 37 C.F.R. § 42.6(e) and 37 C.F.R. § 42.105, **Petition for *Inter Partes* Review of U.S. Patent No. 8,190,610, the associated Power of Attorney, and Exhibits 1001- 1013, 1016-1018, and 1020-1026** were served via Priority Mail Express on the correspondence address of record with the USPTO as follows:

Weaver Austin Villeneuve & Sampson
P.O. Box 70250
Oakland, CA 94612-0250

A courtesy copy is also being sent via FedEx Priority Overnight® to litigation counsel at:

Edward R. Nelson, III
Brent N. Bumgardner
Carder W. Brooks
Christopher G. Granaghan
John P. Murphy
Nelson Bumgardner Conroy PC
3131 W. 7th Street, Suite 300
Fort Worth, TX 76107
Telephone: (817) 377-9111
ed@nelbum.com
brent@nelbum.com
carder@nelbum.com
chris@nelbum.com
murphy@nelbum.com

Dated: March 6, 2023

/s John Campbell
John Campbell (Reg. No. 54,665)