(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 November 2019 (28.11.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/226325 A1

(51) International Patent Classification:
G06F 16/245 (2019.01) *G06F 16/33* (2019.01)

(21) International Application Number:
PCT/US2019/030989

(22) International Filing Date:
07 May 2019 (07.05.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/675,589 23 May 2018 (23.05.2018) US
16/169,928 24 October 2018 (24.10.2018) US

(71) Applicant: **MICROSOFT TECHNOLOGY LICENSING, LLC** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **OKS, Stanislav A.**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **WRIGHT, Travis Austin**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **DANGE, Jasraj Uday**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **JISAROJITO, Jarupat**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **HUANG, Weiyun**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **PADLEY, Stuart**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **JAYACHANDRAN, Umachandar**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **SAINI, Sahaj**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washing-

(54) Title: SCALE OUT DATA STORAGE AND QUERY FILTERING USING STORAGE POOLS

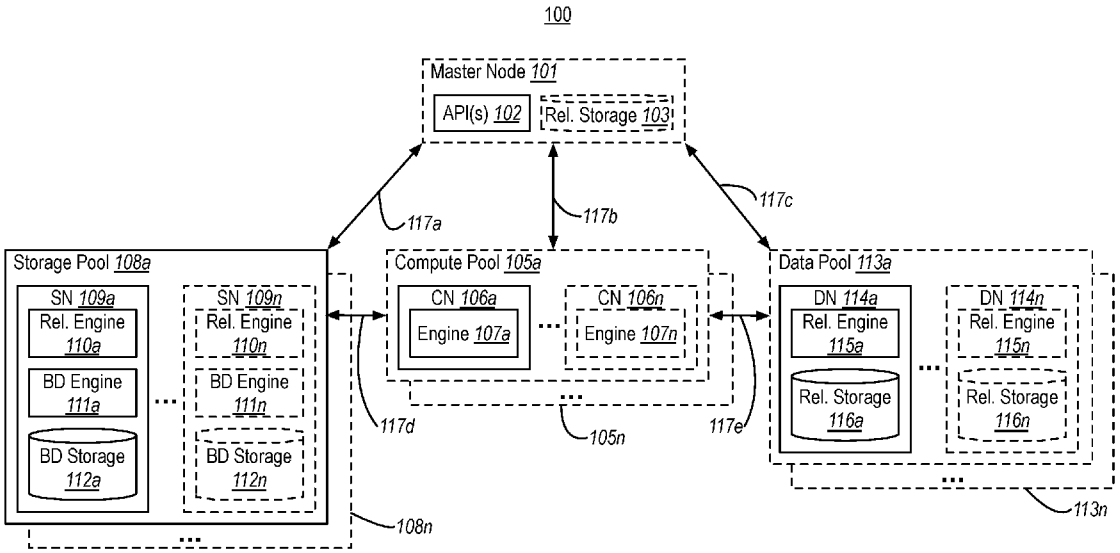

FIG. 1

(57) Abstract: Performing a distributed query across a storage pool includes receiving a database query at a master node or a compute pool within a database system. Based on receiving the database query, a storage pool within the database system is identified. The storage pool comprises a plurality of storage nodes. Each storage node includes a relational engine, a big data engine, and big data storage. The storage pool stores at least a portion of a data set using the plurality of storage nodes by storing a different partition of the data set within the big data storage at each storage node. The database query is processed across the plurality of storage nodes. Query processing includes requesting that each storage node perform a query operation against the partition of the data set stored in its big data storage and return any data from the partition that is produced by the query operation.

WO 2019/226325 A1

[Continued on next page]

ton 98052-6399 (US). **LERCH, William Maxwell**; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US).

(74) **Agent: MINHAS, Sandip S.** et al.; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

SCALE OUT DATA STORAGE AND QUERY FILTERING USING STORAGE POOLS

BACKGROUND

[0001] Computer systems and related technology affect many aspects of society. Indeed, the computer system's ability to process information has transformed the way we live and work. Computer systems now commonly perform a host of tasks (e.g., word processing, scheduling, accounting, etc.) that prior to the advent of the computer system were performed manually. For example, computer systems are commonly used to store and process large volumes of data using different forms of databases.

[0002] Databases can come in many forms. For example, one family of databases follow a relational model. In general, data in a relational database is organized into one or more tables (or "relations") of columns and rows, with a unique key identifying each row. Rows are frequently referred to as records or tuples, and columns are frequently referred to as attributes. In relational databases, each table has an associated schema that represents the fixed attributes and data types that the items in the table will have. Virtually all relational database systems use variations of the Structured Query Language (SQL) for querying and maintaining the database. Software that parses and processes SQL is generally known as an SQL engine. There are a great number of popular relational database engines (e.g., MICROSOFT SQL SERVER, ORACLE, MYSQL POSTGRESQL, DB2, etc.) and SQL dialects (e.g., T-SQL, PL/SQL, SQL/PSM, PL/PGSQL, SQL PL, etc.).

[0003] The proliferation of the Internet and of vast numbers of network-connected devices has resulted in the generation and storage of data on a scale never before seen. This has been particularly precipitated by the widespread adoption of social networking platforms, smartphones, wearables, and Internet of Things (IoT) devices. These services and devices tend to have the common characteristic of generating a nearly constant stream of data, whether that be due to user input and user interactions, or due to data obtained by physical sensors. This unprecedented generation of data has opened the doors to entirely new opportunities for processing and analyzing vast quantities of data, and to observe data patterns on even a global scale. The field of gathering and maintaining such large data sets, including the analysis thereof, is commonly referred to as "big data."

[0004] In general, the term "big data" refers to data sets that are voluminous and/or are not conducive to being stored in rows and columns. For instance, such data sets often comprise blobs of data like audio and/or video files, documents, and other types of

unstructured data. Even when structured, big data frequently has an evolving or jagged schema. Traditional relational database management systems (DBMSs), may be inadequate or sub-optimal for dealing with "big data" data sets due to their size and/or evolving/jagged schemas.

[0005] As such, new families of databases and tools have arisen for addressing the challenges of storing and processing big data. For example, APACHE HADOOP is a collection of software utilities for solving problems involving massive amounts of data and computation. HADOOP includes a storage part, known as the HADOOP Distributed File System (HDFS), as well as a processing part that uses new types of programming models, such as MapReduce, Tez, Spark, Impala, Kudu, etc.

[0006] The HDFS stores large and/or numerous files (often totaling gigabytes to petabytes in size) across multiple machines. The HDFS typically stores data that is unstructured or only semi-structured. For example, the HDFS may store plaintext files, Comma-Separated Values (CSV) files, JavaScript Object Notation (JSON) files, Avro files, Sequence files, Record Columnar (RC) files, Optimized RC (ORC) files, Parquet files, etc. Many of these formats store data in a columnar format, and some feature additional metadata and/or compression.

[0007] As mentioned, big data processing systems introduce new programming models, such as MapReduce. A MapReduce program includes a map procedure, which performs filtering and sorting (e.g., sorting students by first name into queues, one queue for each name), and a reduce method, which performs a summary operation (e.g., counting the number of students in each queue, yielding name frequencies). Systems that process MapReduce programs generally leverage multiple computers to run these various tasks in parallel and manage communications and data transfers between the various parts of the system. An example engine for performing MapReduce functions is HADOOP YARN (Yet Another Resource Negotiator).

[0008] Data in HDFS is commonly interacted with/managed using APACHE SPARK, which provides Application Programming Interfaces (APIs) for executing "jobs" which can manipulate the data (insert, update, delete) or query the data. At its core, SPARK provides distributed task dispatching, scheduling, and basic input/output functionalities, exposed through APIs for interacting with external programming languages, such as Java, Python, Scala, and R.

[0009] Given the maturity of, and existing investment in database technology many organizations may desire to process/analyze big data using their existing relational DBMSs,

leveraging existing tools and knowhow. This may mean importing large amounts of data from big data stores (e.g., such as HADOOP's HDFS) into an existing DBMS. Commonly, this is done using custom-coded extract, transform, and load (ETL) programs that extract data from big data stores, transform the extracted data into a form compatible with traditional data stores, and load the transformed data into an existing DBMS.

[0010] The import process requires not only significant developer time to create and maintain ETL programs (including adapting them as schemas change in the DBMS and/or in the big data store), but it also requires significant time—including both computational time (e.g., CPU time) and elapsed real time (e.g., "wall-clock" time)—and communications bandwidth to actually extract, transform, and transfer the data.

[0011] Given the dynamic nature of big data sources (e.g., continual updates from IoT devices), use of ETL to import big data into a relational DBMS often means that the data is actually out of date/irrelevant by the time it makes it from the big data store into the relational DBMS for processing/analysis. Further, use of ETL leads to data duplication, an increased attack surface, difficulty in creating/enforcing a consistent security model (i.e., across the DBMS and the big data store(s)), geo-political compliance issues, and difficulty in complying with data privacy laws, among other problems.

[0012] Further complicating management of DBMSs and big data systems is planning for and adapting to both computational and storage needs. For example, DBMSs are generally vertically grown—i.e., if more compute or storage capacity is needed it is added to a single computer system, or a more capable computer system is provisioned, and the DBMS is manually migrated to that new computer system. Adding in big data storage and analysis leads to further use of computing resources and requires provisioning of entirely separate computing resources.

BRIEF SUMMARY

[0013] At least some embodiments described herein provide for scale out data storage and query filtering using storage pools in a database system. Storage pools enable the database system to incorporate both relational databases and big data databases, including integrating both relational (e.g., SQL) and big data (e.g., APACHE SPARK) database engines, into a single unified system. In embodiments, this unified database system is configured to make use of pools of resources (e.g., computing resources and storage resources) that can be dynamically added and removed in a scale-out manner as needs vary. Further, these pools are configured to perform distributed data storage and processing across partitioned, providing great flexibility and data processing efficiency.

[0014] In some embodiments, systems, methods, and computer program products for performing a distributed query across a storage pool includes receiving a database query at a master node or a compute pool within a database system. Based on receiving the database query, a storage pool within the database system is identified. The storage pool comprises a plurality of storage nodes, each of which includes a relational engine, a big data engine, and big data storage. The storage pool stores at least a portion of a data set using the plurality of storage nodes by storing a different partition of the data set within the big data storage at each storage node. The database query is processed across the plurality of storage nodes. The query processing includes requesting that each storage node perform a query operation against the partition of the data set stored in its big data storage, and return any data from the partition that is produced by the query operation.

[0015] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0017] Figure 1 illustrates an example of a database system that enables scale out data storage and query filtering using storage pools;

[0018] Figure 2A illustrates an example database system that uses a compute pool for distributed query processing over a storage pool;

[0019] Figure 2B illustrates an example database system that uses a compute pool for distributed query processing over a data pool;

[0020] Figure 2C illustrates an example database system that uses a compute pool for distributed query processing over a storage pool and a data pool;

[0021] Figure 2D illustrates an example of a compute pool performing distributed query processing over a storage pool and a data pool in a partitioned manner;

[0022] Figure 3 illustrates an example database system that includes storage nodes that

provide memory caching functionality; and

[0023] Figure 4 illustrates a flow chart of an example method for performing a distributed query across a storage pool.

DETAILED DESCRIPTION

[0024] At least some embodiments described herein provide for scale out data storage and query filtering using storage pools in a database system. Storage pools enable the database system to incorporate both relational databases and big data databases, including integrating both relational (e.g., SQL) and big data (e.g., APACHE SPARK) database engines, into a single unified system. In embodiments, this unified database system is configured to make use of pools of resources (e.g., computing resources and storage resources) that can be dynamically added and removed in a scale-out manner as needs vary. Further, these pools are configured to perform distributed data storage and processing across partitioned, providing great flexibility and data processing efficiency.

[0025] As will be appreciated in view of the disclosure herein, the embodiments described represent significant advancements in the technical fields of databases and big data processing. For example, by supporting big data engines and big data storage (i.e., in storage pools) as well as traditional database engines, the embodiments herein bring traditional database functionality together with big data functionality within a single managed system for the first time, reducing the number of computer systems that need to be deployed and managed. Additionally, since fewer computer systems are needed to manage relational and big data, the need to continuously transfer and convert data between relational and big data systems is eliminated.

[0026] Figure 1 illustrates an example of a database system 100 that enables data analysis to be performed over the combination of relational data and big data, including scale out data storage and query filtering using storage pools. As shown, database system 100 might include a master node 101. If included, the master node 101 is an endpoint that manages interaction of the database system 100 with external consumers (e.g., other computer systems, software products, etc., not shown) by providing API(s) 102 to receive and reply to queries (e.g., SQL queries). As such, master node 101 can initiate processing of queries received from consumers using other elements of database system 100 (e.g., compute pool(s) 105, storage pool(s) 108, and/or data pool(s) 113, which are described later). Based on obtaining results of processing of queries, the master node 101 can send results back to requesting consumer(s).

[0027] In some embodiments, the master node 101 could appear to consumers to be a

standard relational DBMS. Thus, API(s) 102 could be configured to receive and respond to traditional relational queries. In these embodiments, the master node 101 could include a traditional relational DBMS engine. However, in addition, master node 101 might also facilitate big data queries (e.g., SPARK or MapReduce jobs). Thus, API(s) 102 could also be configured to receive and respond to big data queries. In these embodiments, the master node 101 could also include a big data engine (e.g., a SPARK engine). Regardless of whether master node 101 receives a traditional DBMS query or a big data query, the master node 101 is enabled to process that query over a combination of relational data and big data. While database system 100 provides expandable locations for storing DBMS data (e.g., in data pools 113, as discussed below), it is also possible that master node 101 could include its own relational storage 103 as well (e.g., for storing relational data).

[0028] As shown, database system 100 can include one or more compute pools 105 (shown as 105a-105n). If present, each compute pool 105 includes one or more compute nodes 106 (shown as 106a-106n). The ellipses within compute pool 105a indicate that each compute pool 105 could include any number of compute nodes 106 (i.e., one or more compute nodes 106). Each compute node can, in turn, include a corresponding compute engine 107a (shown as 107a-107n).

[0029] In embodiments, the master node 101 can pass a query received at API(s) 102 to at least one compute pool 105 (e.g., arrow 117b). That compute pool (e.g., 105a) can then use one or more of its compute nodes (e.g., 106a-106n) to process the query against storage pools 108 and/or data pools 113 (e.g., arrows 117d and 117e). These compute node(s) 106 process this query using their respective compute engine 107. After the compute node(s) 106 complete processing of the query, the selected compute pool(s) 105 pass any results back to the master node 101. As will be discussed, in some embodiments, compute pools 105 could also be used to execute scripts (e.g., R, Python, etc.) for training and scoring artificial intelligence (AI) and/or machine learning (ML) models.

[0030] In embodiments, by including compute pools 105, the database system 100 can enable compute capacity (e.g., query processing, AI/ML training/scoring, etc.) of the database system 100 to be to be scaled up efficiently (i.e., by adding new compute pools 105 and/or adding new compute nodes 106 to existing compute pools). The database system 100 can also enable compute capacity to be scaled back efficiently (i.e., by removing existing compute pools 105 and/or removing existing compute nodes 106 from existing compute pools). This enables the database system 100 to scale-out its compute capacity horizontally by provisioning new compute nodes 106 (e.g., physical hardware, virtual

machines, containers, etc.). As such, database system 100 can quickly and efficiently expand or contract its compute capacity as compute demands (e.g., query volume and/or complexity, AI/ML training/scoring demands, etc.) vary.

[0031] In embodiments, if the database system 100 lacks compute pool(s) 105, then the master node 101 may itself handle query processing against storage pool(s) 108, data pool(s) 113, and/or its local relational storage 103 (e.g., arrows 117a and 117c). In embodiments, if one or more compute pools 105 are included in database system 100, these compute pool(s) could be exposed to external consumers directly. In these situations, an external consumer might bypass the master node 101 altogether (if it is present), and initiate queries on those compute pool(s) directly. As such, it will be appreciated that the master node 101 could potentially be optional. If the master node 101 and compute pool(s) 105 are both present, the master node 101 might receive results from each compute pool 105 and join/aggregate those results to form a complete result set.

[0032] As shown, database system 100 includes one or more storage pools 108 (shown as 108a-108n). Each storage pool 108 includes one or more storage nodes 109 (shown as 109a-109n). The ellipses within storage pool 108a indicate that each storage pool could include any number of storage nodes (i.e., one or more storage nodes). As shown, each storage node 109 includes a corresponding relational engine 110 (shown as 110a-110n), a corresponding big data engine 111 (shown as 111a-111n), and corresponding big data storage 112 (shown as 112a-112n). For example, the big data engine 111 could be a SPARK engine, and the big data storage 112 could be HDFS storage. Since storage nodes 109 include big data storage 112, data can be stored at storage nodes 109 using "big data" file formats (e.g., CSV, JSON, etc.), rather than more traditional relational or non-relational database formats. In general, each storage node 109 in storage pool 108 can store a different partition of a big data set.

[0033] Notably, storage nodes 109 in each storage pool 108 can include both a relational engine 110 and a big data engine 111. These engines 110, 111 can be used—singly or in combination—to process queries against big data storage 112 using relational database queries (e.g., SQL queries) and/or using big data queries (e.g., SPARK queries). Thus, the storage pools 108 allow big data to be natively queried with a relational DBMS's native syntax (e.g., SQL), rather than requiring use of big data query formats (e.g., SPARK). For example, storage pools 108 could permit queries over data stored in HDFS-formatted big data storage 112, using SQL queries that are native to a relational DBMS.

[0034] This means that big data can be queried/processed without the need to write

custom tasks (e.g., ETL programs)—making big data analysis fast and readily accessible to a broad range of DBMS administrators/developers. Further, because storage pools 108 enable big data to reside natively within database system 100, they eliminate the need to use ETL techniques to import big data into a DBMS, eliminating the drawbacks described in the Background (e.g., maintaining ETL tasks, data duplication, time/bandwidth concerns, security model difficulties, data privacy concerns, etc.).

[0035] By including storage pools 108, the database system 100 can enable big data storage and processing capacity of the database management system 100 to be scaled up efficiently (i.e., by adding new storage pools 108 and/or adding new storage nodes 109 to existing storage pools). The database system 100 can also enable big data storage and processing capacity to be scaled back efficiently (i.e., by removing existing storage pools 108 and/or removing existing storage nodes 109 from existing storage pools). This enables the database management system 100 to scale-out its big data storage and processing capacity horizontally by provisioning new storage nodes 109 (e.g., physical hardware, virtual machines, containers, etc.). As such, database management system 100 can quickly and efficiently expand or contract its big data storage and processing capacity as the demands for big data storage capacity and processing varies.

[0036] As shown, database system 100 can also include one or more data pools 113 (shown as 113a-113n). If present, each data pool 113 includes one or more data nodes 114 (shown as 114a-114n). The ellipses within data pool 113a indicate that each data pool could include any number of data nodes (i.e., one or more data nodes). As shown, each data node 113 includes a corresponding relational engine 115 (shown as 115a-115n) and corresponding relational storage 116 (shown as 116a-116n). Thus, data pools 113 can be used to store and query relational data stores, where the data is partitioned across individual relational storage 116 within each data node 113.

[0037] Similar to storage pools 103, by including data pools 113 the database system 100 can enable relational storage and processing capacity of the database management system 100 to be scaled up efficiently (i.e., by adding new data pools 113 and/or adding new data nodes 114 to existing data pools). The database system 100 can also enable relational storage and processing capacity to be scaled back efficiently (i.e., by removing existing data pools 113 and/or removing existing data nodes 114 from existing data pools). This enables the database management system 100 to scale-out its relational data storage and processing capacity horizontally by provisioning new data nodes 113 (e.g., physical hardware, virtual machines, containers, etc.). As such, database management system 100 can quickly and

efficiently expand or contract its relational storage and processing capacity as the demands for relational data storage and processing capacity varies.

[0038] Using the relational storage 103, storage pools 108, and/or data pools 113, the database system 100 might be able to process a query (whether that be a relational query or a big data query) over a combination of relational data and big data. Thus, for example, a single query can be processed (e.g., by master node 101 and/or compute pools 105) over any combination of (i) relational data stored at the master node 101 in relational storage 103, (ii) big data stored in big data storage 112 at one or more storage pools 108, and (iii) relational data stored in relational storage 116 at one or more data pools 113. This may be accomplished, for example, by the master node 101 and/or the compute pools 105 creating an "external" table over any data stored at relational storage 103, big data storage 112, and/or relational storage 116. In embodiments, an external table is a logical table that represents a view of data stored in these locations. A single query, sometimes referred to as a global query, can then be processed against a combination of these external tables.

[0039] As mentioned in connection with compute pools 106, database system 100 may execute scripts (e.g., R, Python, etc.) for training and scoring AI and/or ML models based on data stored in database system 100. Similar to how database system 100 enables a query to be run over a combination of relational and big data, database system 100 can also enable such scripts to be run over the combination of relational and big data to train these AI/ML models. Once an AI/ML model is trained, scripts can also be used to "score" the model. In the field of ML, scoring (also called prediction) is the process of new generating values based on a trained ML model, given some new input data. These newly generated values can be sent to an application that consumes ML results or can be used to evaluate the accuracy and usefulness of the model.

[0040] Figures 2A-2D illustrates example database systems 200a-200d in which one or more compute pools 205 are used to perform query (or script) processing across data stored at storage pools 208 and/or data pools 213. The numerals (and their corresponding elements) in Figures 2A-2D correspond to similar numerals (and corresponding elements) from Figure 1. For example, compute pool 205a corresponds to compute pool 105a, storage pool 208a corresponds to storage pool 108a, and so on. As such, all of the description of database system 100 of Figure 1 applies to database systems 200a-200d of Figures 2A-2D. Likewise, all of the additional description of database systems 200a-200d of Figures 2A-2D could be applied to database system 100 of Figure 1.

[0041] In Figures 2A-2D, one or more of the compute pools 205 can receive one or more

queries/scripts from master node 201 and/or from an external consumer. Based on receipt of a query/script, a compute pool 205a can use its compute nodes 206 to execute one or more queries against one or more of the storage pools 208 and/or one or more of the data pools 213. In some embodiments, these queries could be executed in a parallel and distributed manner by the compute nodes 206, as detailed below.

[0042] For example, in Figure 2A, database system 200a includes at least one compute pool 205a and at least one storage pool 208a. As shown by arrows 217f and 217h, each compute node 206 in compute pool 205a could query one or more storage nodes 209 in one or more storage pools 208. In some embodiments, this may include the compute engines 207 at the compute nodes 206 coordinating with the relational engines 210 and/or big data engines 211 at the storage nodes 209. This coordination could include, for example, each compute engine 207 requesting that a relational engine 210 and/or big data 211 engine at a storage node 209 execute an operation across its corresponding partition of a data set stored in its big data storage 212.

[0043] In Figure 2B, on the other hand, database system 200b includes at least one compute pool 205a and at least one data pool 213a. As shown by arrows 217g and 217i, each compute node 206 in compute pool 205a could query one or more data nodes 214 in one or more data pools 213. In some embodiments, this may include the compute engines 207 at the compute nodes 206 coordinating with the relational engines 215 at the data nodes 214. This coordination could include, for example, each compute engine 207 requesting that a relational engine 215 at a data node 214 execute an operation across its corresponding partition of a data set stored in its relational storage 216.

[0044] In Figure 2C database system 200c includes a compute pool 205a as well as both a storage pool 208a and a data pool 213a. As shown by arrows 217f and 217h, each compute node 206 in compute pool 205a might query one or more storage nodes 209 in one or more storage pools 208. In some embodiments, this may include the compute engines 207 at the compute nodes 206 coordinating with the relational engines 210 and/or big data engines 211 at the storage nodes 209. This coordination could include, for example, each compute engine 207 requesting that a relational engine 210 and/or big data 211 engine at a storage node 209 execute an operation across its corresponding partition of a data set stored in its big data storage 212. Likewise, as shown by arrows 217g and 217i, each compute node 206 in compute pool 205a might additionally, or alternatively, query one or more data nodes 214 in one or more data pools 213. In some embodiments, this may include the compute engines 207 at the compute nodes 206 coordinating with the relational engines 215 at the data nodes

214. This coordination could include, for example, each compute engine 207 requesting that a relational engine 215 at a data node 214 execute an operation across its corresponding partition of a data set stored in its relational storage 216.

[0045] It is noted that, for brevity, each compute node 206 is illustrated in Figure 2C as querying both a storage node and a data node. It will be appreciated, however, that in embodiments a compute node 206 may query only storage node(s) 209 or only data node(s) 214. For example, there could be four compute nodes in compute pool 205a, with two of the compute nodes querying respective storage nodes 209, and the other two compute nodes querying respective data nodes 214. In an alternate example, there could be two compute pools—such as compute pools 205a and 205n. In this example, compute nodes in compute pool 205a might query respective storage nodes 209, while compute nodes in compute pool 205n might query respective data nodes 214. Variations of these two examples are also possible.

[0046] In Figures 2A-2C, example operations requested by compute nodes 206 could be filter operations (e.g., a "WHERE" clause in an SQL query), column projection operations, aggregation operations (e.g., local aggregates, partial aggregation), join operations (e.g., partial joins), and the like. Each storage node 209 and/or data node 214 executes a requested operation across its partition of data, and passes any data stored at the node that is produced by the operation back up to the requesting compute node 206. In embodiments, once the compute nodes 206 in each compute pool 205 have received their corresponding portions of results from the various storage/data nodes, they operate together to aggregate/assemble this data in order to form one or more results for the original query/script. Each compute pool 205 then passes these result(s) back to the requesting master node 201 and/or external consumer.

[0047] Figure 2D provides a more concrete example of compute pools 205 receiving corresponding portions of results from partitioned data. In particular, Figure 2D illustrates a database management system 200d, which is generally the same as database management system 200c of Figure 2C, but in which the big data storage 212 and relational storage 216 have been visually expanded to show that there could be different partitions 218 (shown as 218a-218d) of one or more data sets that are stored at the big data storage 212 and/or at the relational storage 216. While the example of Figure 2D (which continues the example, of Figure 2C) illustrates a query across both storage pools 208 and data pools 213, it will be appreciated that the same concepts apply to queries across storage pools only (e.g., Figure 2A) and/or to queries across data pools only (e.g., Figure 2B).

[0048] In view of the description of Figure 2C, it will be appreciated that compute nodes 206 of compute pool 205a could have requested that the storage nodes 209 of storage pool 208a and data nodes 214 of data pool 213a perform one or more operations (e.g., a filter operation) as part of a query on one or more data sets. As shown in Figure 2D, based on having performed these operation(s), some of these nodes could have identified matching portions of data. For example, storage nodes 209a and 209n could have identified data portions 219a and 219b in partitions 218a and 218b, and data node 215a could have identified data portion 219c in partition 218c. Notably, data node 214n has not identified matching data within is corresponding partition 218d. The matching data portions 219a-219c are shown in different sizes to emphasize the matched data could be different at each node, since the nodes store different partitions of a data set. As shown by arrows 217j-217l, the nodes having matching data could pass this data back to the requesting compute nodes 206 in compute pool 205a. These compute nodes 206 can then aggregate/assemble this data to form a final result, which is passed back to the master node 201 and/or a requesting external consumer.

[0049] While Figures 2A-2D have illustrated embodiments in which compute pools 205 are present, it will be appreciated that queries can be distributed across storage pools 208 and/or data pools 213 even when compute pools 205 are not present. For example, master node 201 might directly query one or more storage nodes 209 and/or one or more data nodes 214. In some embodiments, there could even be more than one master node 201, and these plural master nodes could each directly query one or more storage nodes 209 and/or one or more data nodes 214.

[0050] Some embodiments can provide memory caching capabilities to help improve query performance. For example, Figure 3 illustrates example database system 300 that includes storage nodes that provide memory caching functionality. The numerals (and their corresponding elements) in Figure 3 correspond to similar numerals (and corresponding elements) from Figure 1. For example, compute pool 305a corresponds to compute pool 105a, storage pool 308a corresponds to storage pool 108a, and so on. As such, all of the description of database system 300 of Figure 1 applies to database system 300 of Figure 3. Likewise, all of the additional description of database system 300 of Figure 3 could be applied to database system 100 of Figure 1.

[0051] As shown in Figure 3, one or more of the engines in the storage nodes 309 can include cache portions—for example, cache portions 318a-3018n in the relational engines 310 and/or cache portions 318a'-318n' in the big data engines 311. In embodiments, these

cache portions (referred to collectively as cache portions 318) include portions of data that have been queried (e.g., by relational engines 310 and/or big data engines 311) from big data storage 312. Thus, for example, cache portions can include portions of "big data" that have been most recently and/or most frequently accessed from big data storage 312. In embodiments, the cache portions are duplicated across the storage nodes 309. Thus, for example, cache portions 318a in relational engine 310a might be the same as cache portions 318n in relational engine 310n. These cache portions 318 can be used to improve performance of queries against big data storage 312.

[0052] While the foregoing description has focused on example systems, embodiments herein can also include methods that are performed within those systems. Figure 4, for example, illustrates a flow chart of an example method 400 for performing a distributed query across a storage pool. In embodiments, method 400 could be performed, for example, within database systems 100, 200a-200d, and/or 300 of Figures 1-3.

[0053] As shown, method 400 includes an act 401 of receiving a database query. In some embodiments, act 401 comprises receiving a database query at a master node or a compute pool within a database system. For example, as was discussed in connection with Figure 1, database system 100 could include a relational master node 101. If so, this relational master node 101 could receive a database query from an external consumer. Thus, act 401 could comprise the database query being received at the master node. Additionally, or alternatively, database system 100 could include one or more compute pools 105, each including one or more compute nodes. If database system 100 includes both a master node 101 and a compute pool 105, act 401 could comprise the database query being received at the master node 101, and the master node 101 passing the database query to the compute pool 105. Alternatively, act 401 could comprise the database query being received at the compute pool 105 directly (whether or not master node 101 is present). For example, as was discussed in connection with Figure 1, external consumers might be made aware of compute pool(s) 105 and might be enabled to query them directly.

[0054] Method 400 also includes an act 402 of identifying a storage pool. In some embodiments, act 402 comprises, based on receiving the database query, identifying a storage pool within the database system. In act 402, the storage pool could comprise a plurality of storage nodes, each storage node including a relational engine, a big data engine, and big data storage. The storage pool could also store at least a portion of a data set using the plurality of storage nodes by storing a different partition of the data set within the big data storage at each storage node. For example, if the database query was received at the

master node 101, then the master node 101 might identify storage pool 108a. In another example, the database query might have been received at master node 101 and passed to compute pool 105a, in which case compute pool 105a could identify storage pool 108a. In yet another example, the database query could have been received by compute pool 105a directly, in which case compute pool 105a could identify storage pool 108a.

[0055] Method 400 also includes an act 403 of processing the database query across a plurality of storage nodes. In some embodiments, act 403 comprises processing the database query across the plurality of storage nodes, including requesting that each storage node perform a query operation against the partition of the data set stored in its big data storage, and return any data from the partition that is produced by the query operation. For example, master node 101 could query each storage node 109 of a storage pool 108. As such, act 403 could comprise the master node processing the database query across the plurality of storage nodes.

[0056] Additionally, or alternatively, compute nodes 106 of compute pool 105a could query each storage node 109 of a storage pool 108. Specific examples of querying by a compute pool are shown in Figures 2A and 2C. As such, act 403 could comprise the compute pool processing the database query across the plurality of storage nodes. For example, as shown by arrows 217f and 217h, compute node 206a could query storage node 209a, and compute node 206n could query storage node 209n. Figure 2D shows that, based on this querying, the storage nodes can return results (i.e., as indicated by arrows 217j and 217l). From the discussion of Figures 2A and 2C, it is clear that, when querying is performed by a compute pool, act 403 could comprise the compute pool processing the database query across the plurality of storage nodes by using a different compute node to query each storage node.

[0057] When a storage node performs a query operation against the partition of the data set stored in its big data storage, it could use one or both of its relational engine 110 or its big data engine 111. Thus, method 400 could include one or more of (i) each storage node performing the query operation against the partition of the data set stored in its big data storage using its relational engine, or (ii) each storage node performing the query operation against the partition of the data set stored in its big data storage using its big data engine. The particular query operation(s) performed can vary depending on the original database query, but examples include at least one of a filter operation, a column projection operation, an aggregation operation, or a join operation.

[0058] Method 400 need not be limited to querying storage nodes. For example, as

shown in Figure 1, database system 100 could also include one or more data pools. As such, the computer system performing method 400 could also comprise a data pool comprising a plurality of data nodes, each data node comprising a relational engine and a relational data storage. In these embodiments, the computer system can also process the database query across the plurality of data nodes, including requesting that each data node perform a query operation against a partition of the data set stored in its relational storage, and return any data from the partition that is produced by the query operation.

[0059] As was discussed, a compute pool can aggregate results received from storage nodes and/or data nodes. For example, referring to Figure 2D, once compute nodes 206a and 206n receive data portions 219a and 219b from storage nodes 209a and 209n, compute nodes 206a and 206n can aggregate those data portions 219. In the particular example of Figure 2D, which includes data pool 213a, compute nodes 206a and 206n and also aggregate data potion 219c received from data node 214a. Thus, method 400 can also include the compute pool aggregating results received by each compute node (i.e., from storage nodes and/or data nodes).

[0060] As was discussed, compute pools 105, storage pools 108, and data pools 113 enable database system 100 to dynamically expand and contract its compute capacity, its big data storage and processing capacity, and/or its relational storage and processing capacity. Thus, the computer system performing method 400 could expand its compute capacity by adding one or more compute nodes, could expand its big data storage capacity by adding one or more storage nodes, and/or could expand its relational storage capacity by adding one or more data nodes. Any of these capacities could be contracted be removing respective nodes.

[0061] Also, as discussed in connection with Figure 3, a storage pool might use its storage nodes to cache result portions in their relational and/or big data engines (e.g., in a memory cache). Thus, in method 400, each storage node could store a set of cache portions that comprises data that has been accessed from the big data storage at one or more of the plurality of storage nodes.

[0062] Accordingly, the embodiments herein provide for scale out data storage and query filtering using storage pools in a database system. As was discussed, storage pools enable the database system to incorporate both relational databases and big data databases, including integrating both relational (e.g., SQL) and big data (e.g., APACHE SPARK) database engines, into a single unified system. This unified database system makes use of pools of resources (e.g., computing resources and storage resources) that can be dynamically

added and removed in a scale-out manner as needs vary. Further, these pools are configured to perform distributed data storage and processing across partitioned, providing great flexibility and data processing efficiency.

[0063] It will be appreciated that embodiments of the present invention may comprise or utilize a special-purpose or general-purpose computer system that includes computer hardware, such as, for example, one or more processors and system memory, as discussed in greater detail below. Embodiments within the scope of the present invention also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general-purpose or special-purpose computer system. Computer-readable media that store computer-executable instructions and/or data structures are computer storage media. Computer-readable media that carry computer-executable instructions and/or data structures are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: computer storage media and transmission media.

[0064] Computer storage media are physical storage media that store computer-executable instructions and/or data structures. Physical storage media include computer hardware, such as RAM, ROM, EEPROM, solid state drives ("SSDs"), flash memory, phase-change memory ("PCM"), optical disk storage, magnetic disk storage or other magnetic storage devices, or any other hardware storage device(s) which can be used to store program code in the form of computer-executable instructions or data structures, which can be accessed and executed by a general-purpose or special-purpose computer system to implement the disclosed functionality of the invention.

[0065] Transmission media can include a network and/or data links which can be used to carry program code in the form of computer-executable instructions or data structures, and which can be accessed by a general-purpose or special-purpose computer system. A "network" is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer system, the computer system may view the connection as transmission media. Combinations of the above should also be included within the scope of computer-readable media.

[0066] Further, upon reaching various computer system components, program code in

the form of computer-executable instructions or data structures can be transferred automatically from transmission media to computer storage media (or vice versa). For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a "NIC"), and then eventually transferred to computer system RAM and/or to less volatile computer storage media at a computer system. Thus, it should be understood that computer storage media can be included in computer system components that also (or even primarily) utilize transmission media.

[0067] Computer-executable instructions comprise, for example, instructions and data which, when executed at one or more processors, cause a general-purpose computer system, special-purpose computer system, or special-purpose processing device to perform a certain function or group of functions. Computer-executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code.

[0068] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, tablets, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. As such, in a distributed system environment, a computer system may include a plurality of constituent computer systems. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

[0069] Those skilled in the art will also appreciate that the invention may be practiced in a cloud computing environment. Cloud computing environments may be distributed, although this is not required. When distributed, cloud computing environments may be distributed internationally within an organization and/or have components possessed across multiple organizations. In this description and the following claims, "cloud computing" is defined as a model for enabling on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). The definition of "cloud computing" is not limited to any of the other numerous advantages that can be obtained from such a model when properly deployed.

[0070] A cloud computing model can be composed of various characteristics, such as on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service, and so forth. A cloud computing model may also come in the form of various service models such as, for example, Software as a Service ("SaaS"), Platform as a Service ("PaaS"), and Infrastructure as a Service ("IaaS"). The cloud computing model may also be deployed using different deployment models such as private cloud, community cloud, public cloud, hybrid cloud, and so forth.

[0071] Some embodiments, such as a cloud computing environment, may comprise a system that includes one or more hosts that are each capable of running one or more virtual machines. During operation, virtual machines emulate an operational computing system, supporting an operating system and perhaps one or more other applications as well. In some embodiments, each host includes a hypervisor that emulates virtual resources for the virtual machines using physical resources that are abstracted from view of the virtual machines. The hypervisor also provides proper isolation between the virtual machines. Thus, from the perspective of any given virtual machine, the hypervisor provides the illusion that the virtual machine is interfacing with a physical resource, even though the virtual machine only interfaces with the appearance (e.g., a virtual resource) of a physical resource. Examples of physical resources including processing capacity, memory, disk space, network bandwidth, media drives, and so forth.

[0072] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

1. A computer system, comprising:

one or more processors; and

one or more computer-readable media having stored thereon computer-executable instructions, that when executed at the one or more processors, cause the computer system to perform the following:

receive a database query at a master node or a compute pool within a database system;

based on receiving the database query, identify a storage pool within the database system, in which,

the storage pool comprises a plurality of storage nodes, each storage node including a relational engine, a big data engine, and big data storage; and

the storage pool stores at least a portion of a data set using the plurality of storage nodes by storing a different partition of the data set within the big data storage at each storage node; and

process the database query across the plurality of storage nodes, including requesting that each storage node perform a query operation against the partition of the data set stored in its big data storage, and return any data from the partition that is produced by the query operation.

2. The computer system as recited in claim 1, wherein the database query is received at the master node, and wherein the master node processes the database query across the plurality of storage nodes.

3. The computer system as recited in claim 1, wherein the database query is received at the master node, and wherein the master node passes the database query to the compute pool, which processes the database query across the plurality of storage nodes.

4. The computer system as recited in claim 1, wherein the database query is received at the compute pool, and wherein the compute pool processes the database query across the plurality of storage nodes.

5. The computer system as recited in claim 4, wherein the compute pool processes the database query across the plurality of storage nodes by using a different compute node to query each storage node.

6. The computer system as recited in claim 5, wherein the compute pool aggregates results received by each compute node.

7. The computer system as recited in claim 1, wherein each storage node performs the query operation against the partition of the data set stored in its big data storage using its relational engine.

8. The computer system as recited in claim 1, wherein each storage node performs the query operation against the partition of the data set stored in its big data storage using its big data engine.

9. The computer system as recited in claim 1, wherein the computer system expands its compute capacity by adding one or more compute nodes.

10. The computer system as recited in claim 1, wherein the computer system expands its big data storage capacity by adding one or more storage nodes.

11. The computer system as recited in claim 1, wherein the computer system also comprises a data pool comprising a plurality of data nodes, each data node comprising a relational engine and a relational data storage.

12. The computer system as recited in claim 11, wherein the computer system also processes the database query across the plurality of data nodes, including requesting that each data node perform a query operation against a partition of the data set stored in its relational storage, and return any data from the partition that is produced by the query operation.

13. The computer system as recited in claim 1, wherein each storage node stores a set of cache portions that comprises data that has been accessed from the big data storage at one or more of the plurality of storage nodes.

14. The computer system as recited in claim 1, wherein the query operation comprises at least one of a filter operation, a column projection operation, an aggregation operation, or a join operation.

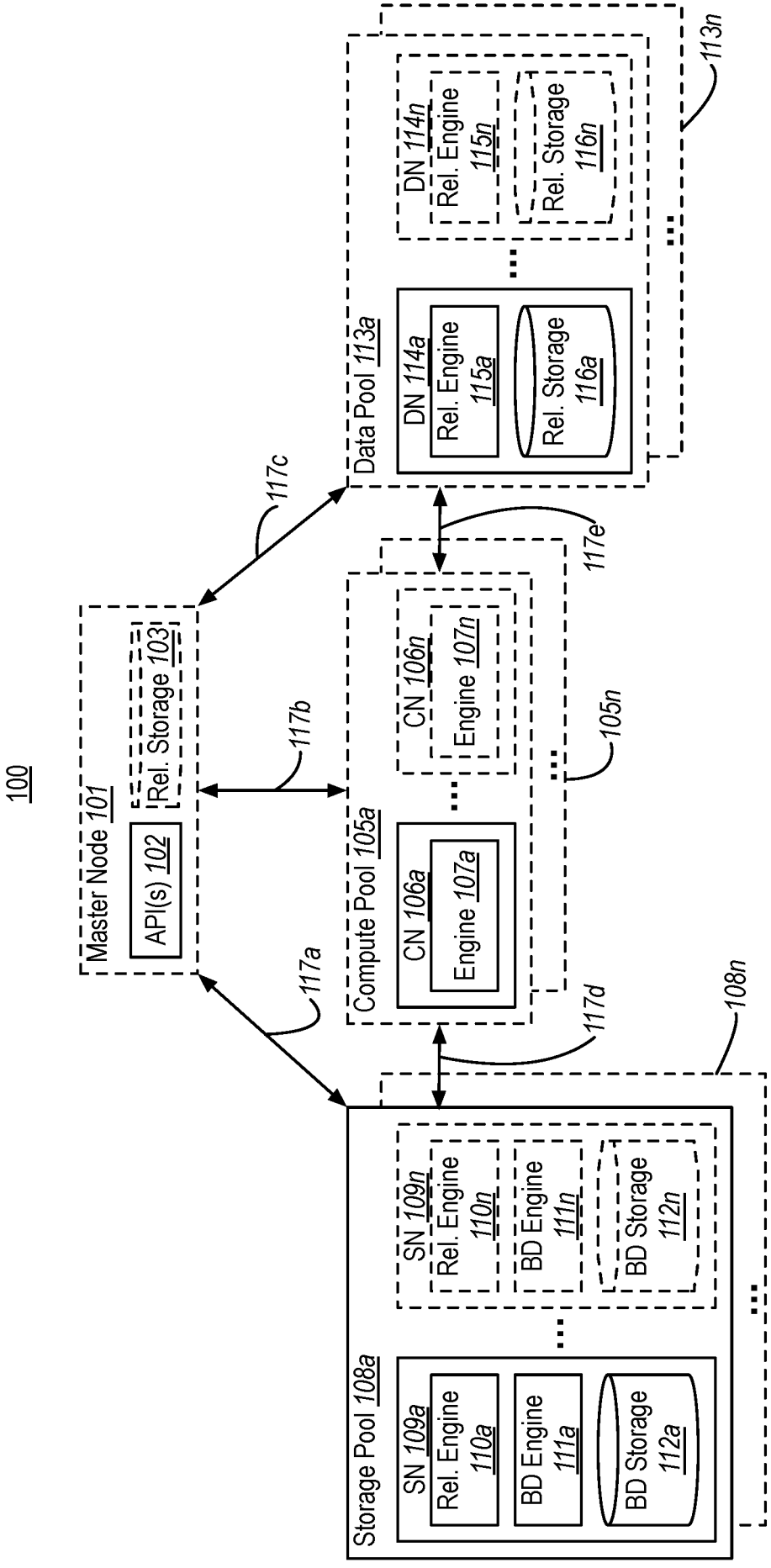
100
Master Node 101
API(s) 102
Rel. Storage 103
117a
117b
117c
117d
117e
Compute Pool 105a
CN 106a
Engine 107a
CN 106n
Engine 107n
105n
Data Pool 113a
DN 114a
Rel. Engine 115a
Rel. Storage 116a
DN 114n
Rel. Engine 115n
Rel. Storage 116n
113n
108n
Storage Pool 108a
SN 109a
Rel. Engine 110a
BD Engine 111a
BD Storage 112a
SN 109n
Rel. Engine 110n
BD Engine 111n
BD Storage 112n

FIG. 1

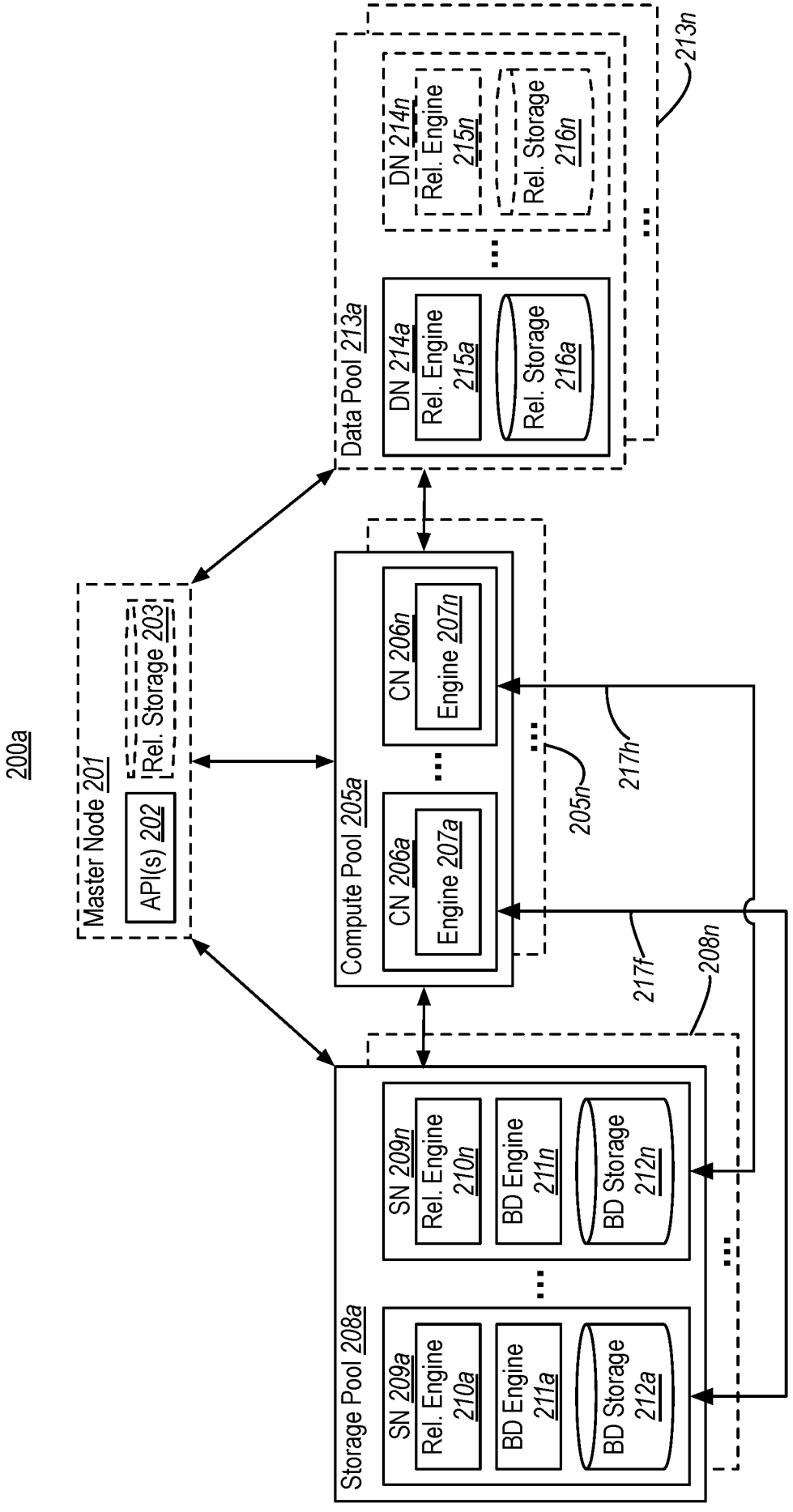
200a
Master Node 201
API(s) 202
Rel. Storage 203
Compute Pool 205a
CN 206a
Engine 207a
CN 206n
Engine 207n
205n
217h
217f
208n
Storage Pool 208a
SN 209a
Rel. Engine 210a
BD Engine 211a
BD Storage 212a
SN 209n
Rel. Engine 210n
BD Engine 211n
BD Storage 212n
Data Pool 213a
DN 214a
Rel. Engine 215a
Rel. Storage 216a
DN 214n
Rel. Engine 215n
Rel. Storage 216n
213n

FIG. 2A

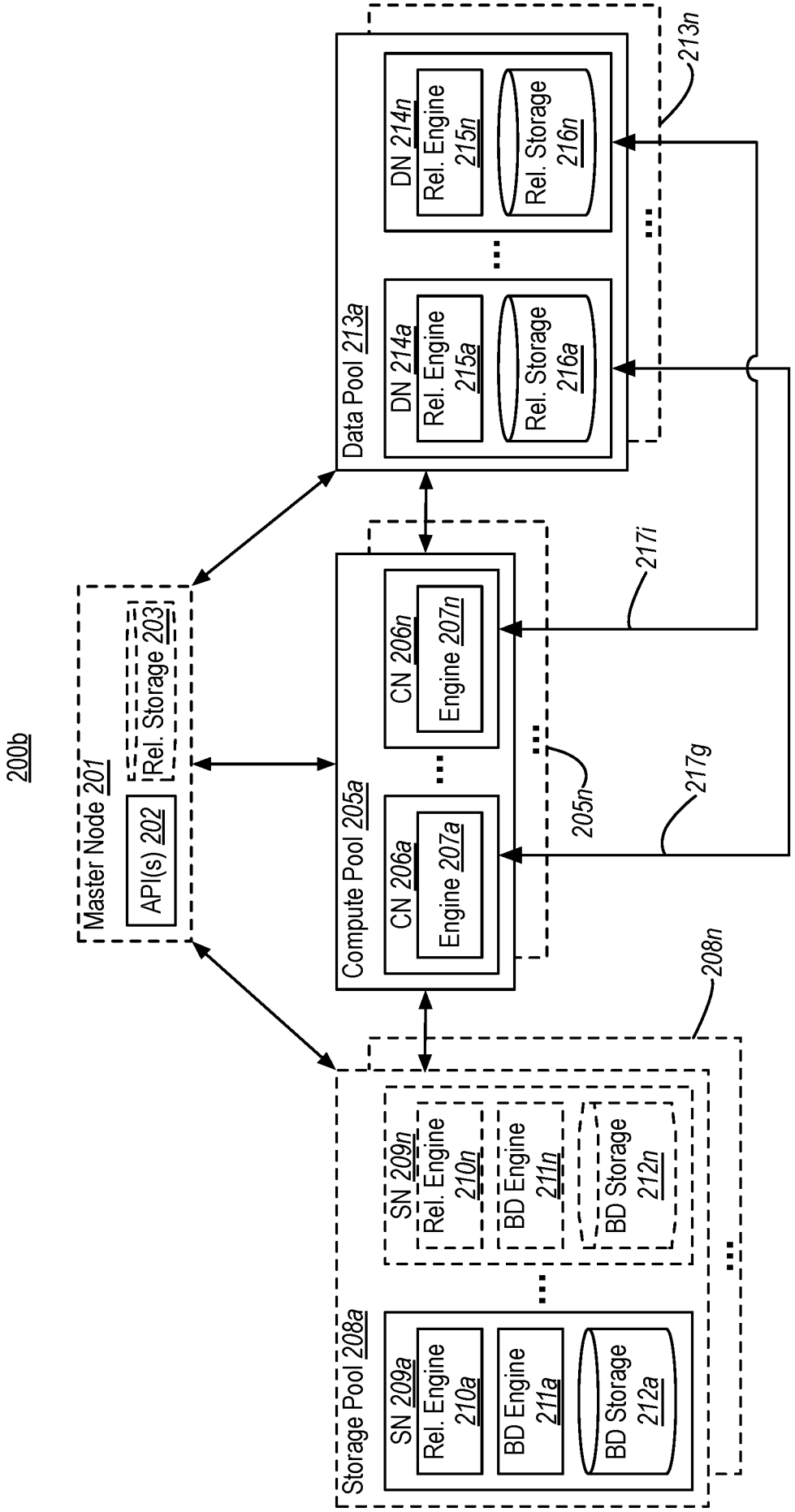
200b
Master Node 201
API(s) 202
Rel. Storage 203
Compute Pool 205a
CN 206a
Engine 207a
CN 206n
Engine 207n
205n
Data Pool 213a
DN 214a
Rel. Engine 215a
Rel. Storage 216a
DN 214n
Rel. Engine 215n
Rel. Storage 216n
213n
217i
217g
208n
Storage Pool 208a
SN 209a
Rel. Engine 210a
BD Engine 211a
BD Storage 212a
SN 209n
Rel. Engine 210n
BD Engine 211n
BD Storage 212n

FIG. 2B

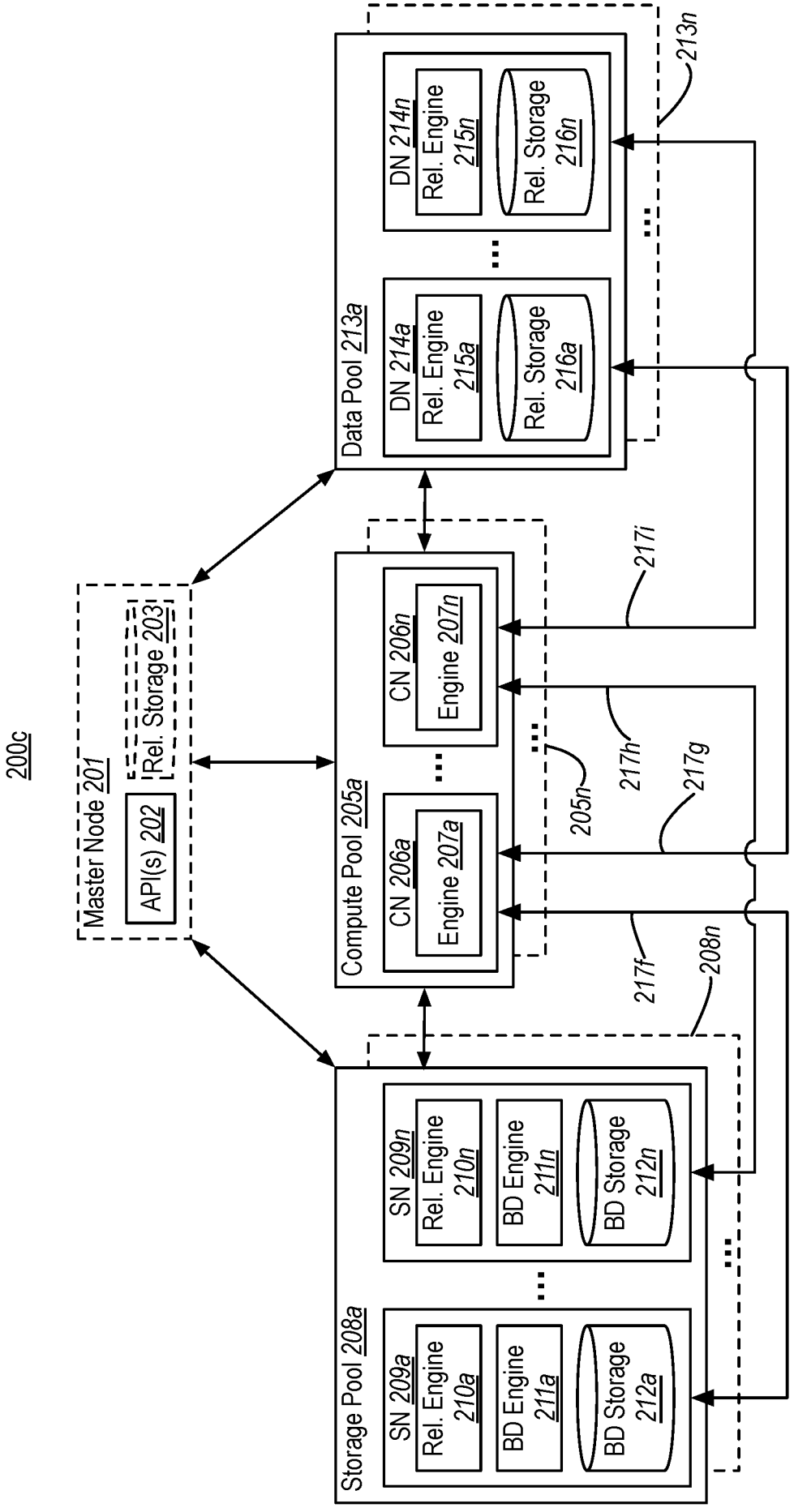
200c
Master Node 201
API(s) 202
Rel. Storage 203
Compute Pool 205a
CN 206a
Engine 207a
CN 206n
Engine 207n
Storage Pool 208a
SN 209a
Rel. Engine 210a
BD Engine 211a
BD Storage 212a
SN 209n
Rel. Engine 210n
BD Engine 211n
BD Storage 212n
Data Pool 213a
DN 214a
Rel. Engine 215a
Rel. Storage 216a
DN 214n
Rel. Engine 215n
Rel. Storage 216n
213n
217i
205n
217h
217g
217f
208n

FIG. 2C

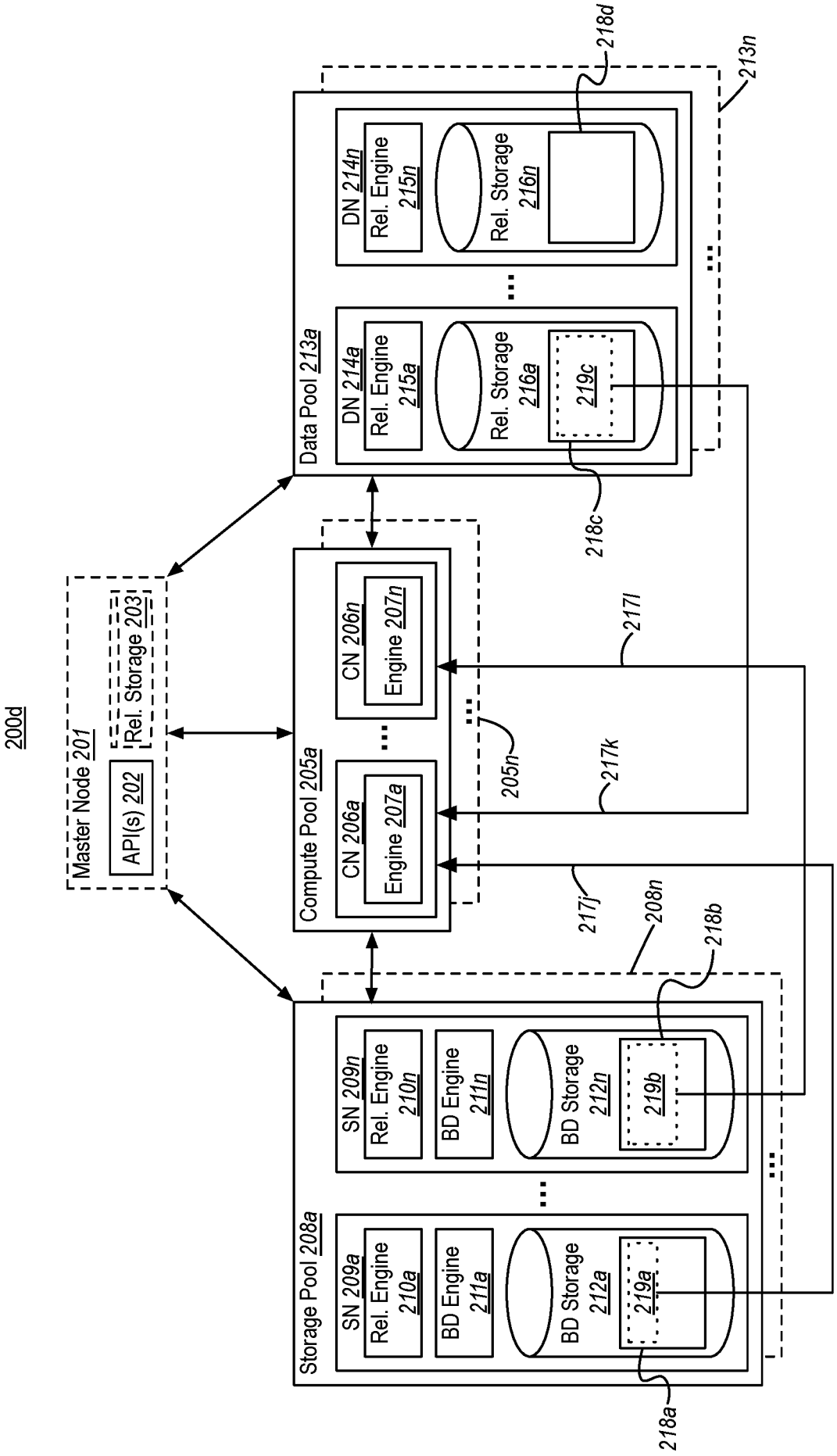
200d
Master Node 201
API(s) 202
Rel. Storage 203
Compute Pool 205a
CN 206a
Engine 207a
CN 206n
Engine 207n
205n
Data Pool 213a
DN 214a
Rel. Engine 215a
Rel. Storage 216a
219c
DN 214n
Rel. Engine 215n
Rel. Storage 216n
213n
Storage Pool 208a
SN 209a
Rel. Engine 210a
BD Engine 211a
BD Storage 212a
219a
SN 209n
Rel. Engine 210n
BD Engine 211n
BD Storage 212n
219b
208n
217j
217k
217l
218a
218b
218c
218d

FIG. 2D

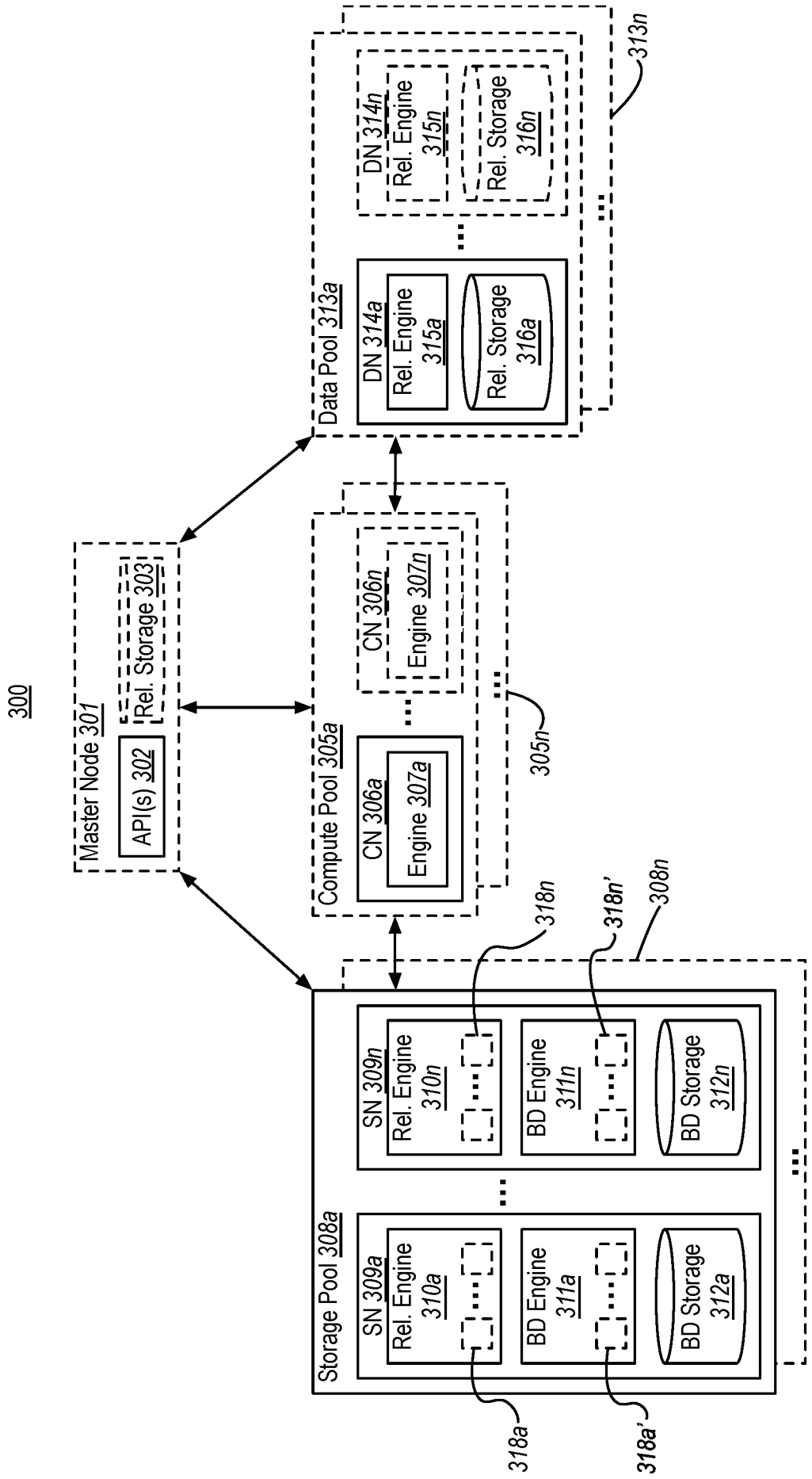
300
Master Node 301
API(s) 302
Rel. Storage 303
Compute Pool 305a
CN 306a
Engine 307a
CN 306n
Engine 307n
305n
Data Pool 313a
DN 314a
Rel. Engine 315a
Rel. Storage 316a
DN 314n
Rel. Engine 315n
Rel. Storage 316n
313n
Storage Pool 308a
SN 309a
Rel. Engine 310a
BD Engine 311a
BD Storage 312a
SN 309n
Rel. Engine 310n
BD Engine 311n
BD Storage 312n
308n
318a
318n
318a'
318n'

FIG. 3

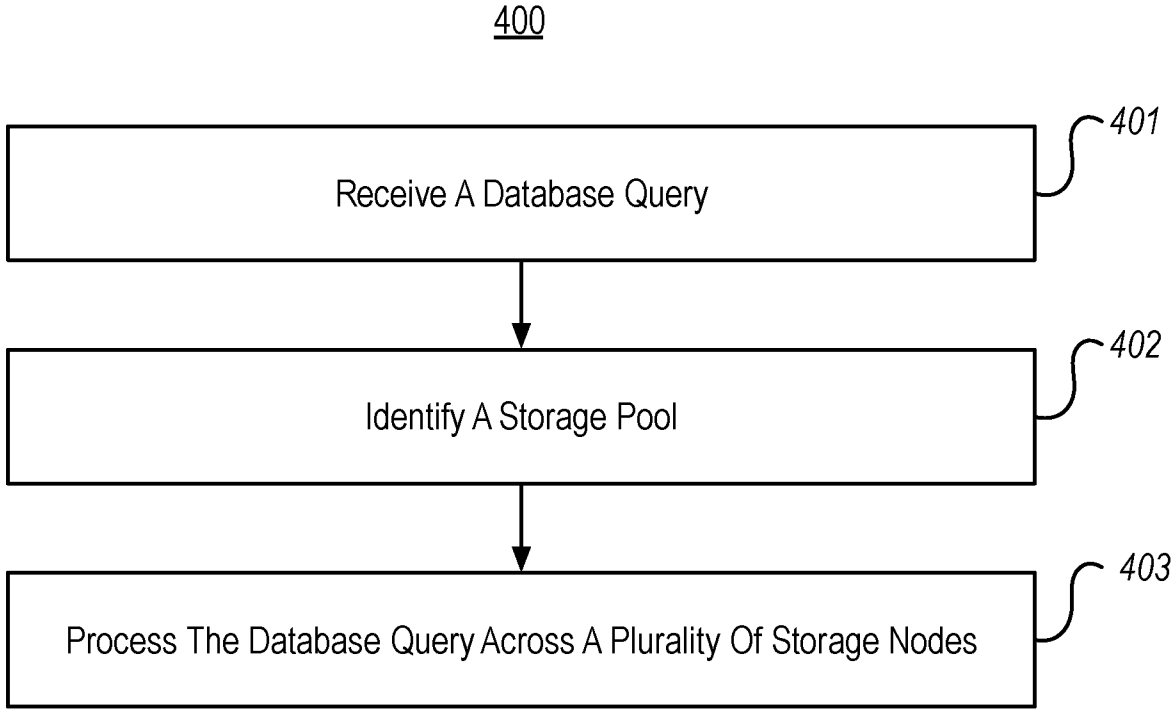
400
Receive A Database Query
401
Identify A Storage Pool
402
Process The Database Query Across A Plurality Of Storage Nodes
403

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2019/030989

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F16/245 G06F16/33
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2012/310916 A1 (ABADI DANIEL [US] ET AL) 6 December 2012 (2012-12-06) paragraph [0060] - paragraph [0068] paragraph [0104] -----	1-14
X	EP 2 778 967 A1 (CLOUDERA INC [US]) 17 September 2014 (2014-09-17) paragraph [0012] paragraph [0020] - paragraph [0021] ----- -/--	1-14

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 July 2019	12/07/2019

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Boubal, François

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	BAGET JEAN-FRANÇOIS ET AL: "ALASKA for Ontology Based Data Access", 26 May 2013 (2013-05-26), INTERNATIONAL CONFERENCE ON COMPUTER ANALYSIS OF IMAGES AND PATTERNS. CAIP 2017: COMPUTER ANALYSIS OF IMAGES AND PATTERNS; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER, BERLIN, HEIDELBERG, PAGE(S) 157 - 161, XP047491054, ISBN: 978-3-642-17318-9 the whole document	1-14
A	TOM J AMELOOT ET AL: "Data partitioning for single-round multi-join evaluation in massively parallel systems", SIGMOD RECORD, ACM, NEW YORK, NY, US, vol. 45, no. 1, 2 June 2016 (2016-06-02), pages 33-40, XP058262012, ISSN: 0163-5808, DOI: 10.1145/2949741.2949750 the whole document	1-14

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2019/030989

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2012310916 A1	06-12-2012	EP 2729883 A2	14-05-2014
		US 2012310916 A1	06-12-2012
		WO 2013009503 A2	17-01-2013
EP 2778967 A1	17-09-2014	AU 2014201441 A1	02-10-2014
		CA 2843459 A1	08-07-2014
		CA 2912038 A1	08-07-2014
		EP 2778967 A1	17-09-2014
		GB 2511935 A	17-09-2014
		JP 6050272 B2	21-12-2016
		JP 2014194769 A	09-10-2014
		KR 20140112427 A	23-09-2014
		US 2014280032 A1	18-09-2014
		US 2017132283 A1	11-05-2017