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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 June 2015 (11.06.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/084760 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2014/067999

(22) International Filing Date:
2 December 2014 (02.12.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/910,841 2 December 2013 (02.12.2013) US

(71) Applicant: **QBASE, LLC** [US/US]; 12018 Sunrise Valley Drive, Suite 300, Reston, VA 20191 (US).

(72) Inventors: **LIGHTNER, Scott**; 22596 Redhill Manor Court, Leesburg, VA 20175 (US). **WECKESSER, Franz**; 3942 E Centerville Road, Spring Valley, OH 45370 (US). **BERKEY, Telford**; 2190 Cherokee Drive, London, OH 43140 (US). **BECKNELL, Joseph**; 3910 Elmira Drive, Kettering, OH 45439 (US). **ZIMMERMAN, Bryan**; 3086 Catlett Road, Catlett, VA 20119 (US). **PERSSON, Mats**; 2034 Cordoba PL, Carlsbad, CA 92008 (US).

(74) Agent: **SOPHIR, Eric**; Dentons US LLP, P.O. Box 061080, Wacker Drive Station, Willis Tower, Chicago, IL 60606 (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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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).

Published:

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

(54) **Title**: DESIGN AND IMPLEMENTATION OF CLUSTERED IN-MEMORY DATABASE

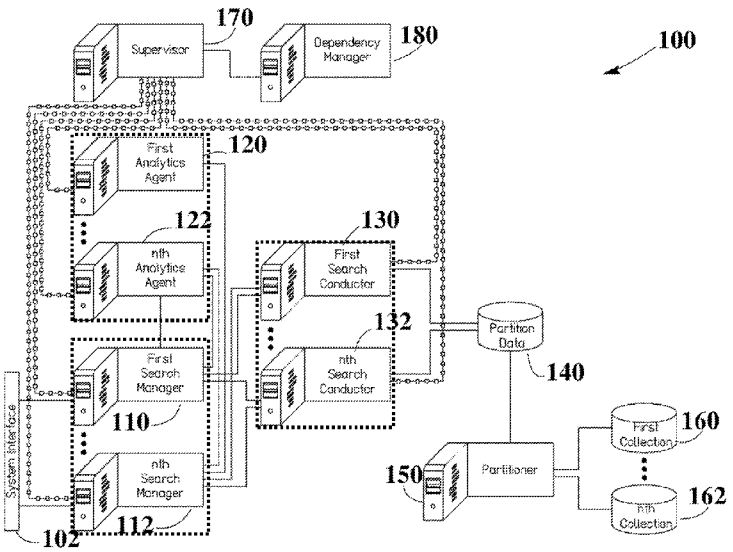

FIG. 1

(57) **Abstract**: An in-memory database system and method for administrating a distributed in-memory database, comprising one or more nodes having modules configured to store and distribute database partitions of collections partitioned by a partitioner associated with a search conductor. Database collections are partitioned according to a schema. Partitions, collections, and records, are updated and removed when requested by a system interface, according to the schema. Supervisors determine a node status based on a heartbeat signal received from each node. Users can send queries through a system interface to search managers. Search managers apply a field processing technique, forward the search query to search conductors, and return a set of result records to the analytics agents. Analytics agents perform analytics processing on a candidate results records from a search manager. The search conductors comprising partitioners associated with a collection, search and score the records in a partition, then return a set of candidate result records after receiving a search query from a search manager.

WO 2015/084760 A1

DESIGN AND IMPLEMENTATION OF CLUSTERED IN-MEMORY DATABASE

TECHNICAL FIELD

[0001] The present disclosure relates in general to databases, and more specifically to in-memory databases.

BACKGROUND

[0002] Computers are powerful tools of use in storing and providing access to vast amounts of information, while databases are a common mechanism for storing information on computer systems while providing easy access to users. Typically, a database is an organized collection of information stored as "records" having "fields" of information (e.g., a restaurant database may have a record for each restaurant in a region, where each record contains fields describing characteristics of the restaurant, such as name, address, type of cuisine, and the like).

[0003] In operation, a database management system frequently needs to retrieve data from or persist data to storage devices such as disks. Unfortunately, access to such storage devices can be somewhat slow. To speed up access to data, databases typically employ a "cache" or "buffer cache" which is a section of relatively faster memory (e.g., random access memory (RAM)) allocated to store recently used data objects. Memory is typically provided on semiconductor or other electrical storage media and is coupled to a CPU (central processing unit) via a fast data bus which enables data maintained in memory to be accessed more rapidly than data stored on disks.

[0004] One approach that may be taken when attempting to solve this problem is to store all the information in the database in memory, however as memory provided on

computer systems has a limited size there are a number of obstacles that must be faced when attempting to handle databases of a larger scale.

[0005] As such, there is a continuing need for improved methods of storing and retrieving data at high speeds at a large scale.

SUMMARY

[0006] Disclosed herein is a system architecture hosting an in-memory database, which may include any suitable combination of computing devices and software modules for storing, manipulating, and retrieving data records of the in-memory database that is hosted within the distributed computing architecture of the system. Software modules executed by computing hardware of the system may include a system interface, a search manager, an analytics agent, a search conductor, a partitioner, collections of data, a supervisor, a dependency manager; any suitable combination of these software modules may be found in the system architecture hosting the in-memory database.

[0007] Nodes executing software modules may compress data stored in the records to make in-memory storage, queries, and retrieval feasible for massive data sets. Compression and decompression may be performed at nearly any level of the database (e.g., database level, collection level, record level, field level). Nodes executing software modules may provide support for storing complex data structures, such as JavaScript Object Notation (JSON) in the distributed in-memory database. Embodiments of an in-memory database system may be fault-tolerant due to the distributed architecture of system components and the various hardware and software modules of the system that are capable of monitoring and restoring faulty services. Fault-tolerance may include system component redundancy, and automated recovery procedures for system components, among other techniques. The in memory database may effectively and efficiently query data by scoring data using scoring methods.

Search results may be ranked according to the scoring methods used to score the data, thereby allowing users and/or nodes executing queries to utilize data in ways that are more tailored and contextually relevant from one query to the next. Nodes executing analytics agents may perform various advanced analytics on records stored in the in-memory database image of data. In some cases, analytics may be performed on the records retrieved with a set of search query results by search conductors.

[0008] In one embodiment, a computing system hosting an in-memory database, the system comprising: a partitioner node comprising a processor configured to, in response to receiving a collection of one or more records of a database, determine whether to compress the collection based on a machine-readable schema file associated with the collection, logically partition the collection into one or more partitions according to the schema file, and distribute the one or more partitions to one or more storage nodes according to the schema file; a storage node comprising non-transitory machine-readable main memory storing a partition received from the partitioner associated with the storage node; a search manager node comprising a processor receiving a search query from a client device of the system, and transmitting the search queries as search conductor queries to one or more search conductors in response to receive the search query from the client device, wherein the search query is a machine-readable computer file containing parameters associated with one or more records satisfying the search query; a search conductor node associated with one or more partitioners and comprising a processor configured to, in response to receiving a search conductor query from the search manager node: query a set of one or more partitions indicated by the search conductor query, identify one or more candidate records stored in the set of queried partitions, calculate a first score for each respective candidate record using a scoring algorithm, and transmit to the search manager a set of one or more query results containing one or more candidate records satisfying a threshold value; and an analytics agent node

comprising a processor configured to automatically generate a machine-readable computer file containing a set of one or more data linkages for the set of query results, responsive to identifying in the set of query results received from the search manager node a data linkage correlating two or more records, wherein the data linkage correlates data contained in a first record associated with data contained in a second record.

[0009] In another embodiment, a computer implemented method comprises receiving, by a search manager computer of a system hosting an in-memory database, binary data representing a search query containing parameters querying the database, wherein the system comprises one or more storage nodes comprising main memory storing one or more collections of the database, wherein each collection contains one or more records, transmitting, by the computer, the search query to one or more search conductor nodes according to the search query, wherein the search query indicates a set of one or more collections to be queried; transmitting, by the computer, to one or more analytics agent nodes a set of search results based on the search query responsive to receiving from the one or more search conductors the set of search results containing one or more records satisfying the search query, wherein each respective record of the set of search results is associated with a score based on a scoring algorithm in the search query; and responsive to the computer receiving a computer file containing a set of one or more data linkages from the one or more analytics agent nodes: updating, by the computer, the one or more records of the set of search results according to the set of one or more data linkages received from the analytics agent nodes.

[0010] In another embodiment, a computer-implemented method comprises receiving, by a computer, one or more collections from a search conductor according to a schema file, wherein each of the collections comprises a set of one or more records having

one or more fields; partitioning, by the computer, each collection according to the schema; compressing, by the computer, the records in the partition according to the schema; and distributing, by the computer, each of the partitions to one or more associated search conductors to include each of the partitions in each collection corresponding to the partitioner associated with the search conductor.

[0011] Numerous other aspects, features of the present disclosure may be made apparent from the following detailed description. Additional features and advantages of an embodiment will be set forth in the description which follows, and in part will be apparent from the description. The objectives and other advantages of the invention will be realized and attained by the structure particularly pointed out in the exemplary embodiments in the written description and claims hereof as well as the appended drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The present disclosure can be better understood by referring to the following figures. The components in the figures are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the disclosure. In the figures, reference numerals designate corresponding parts throughout the different views.

[0013] **FIG. 1** shows an in-memory database architecture according to an exemplary embodiment.

[0014] **FIG. 2** shows a node configuration according to an exemplary embodiment.

[0015] **FIG. 3** is a flow chart for setting up a node according to an exemplary embodiment.

[0016] **FIG. 4** is a flow chart depicting module set up in a node according to an exemplary embodiment.

[0017] **FIG. 5** is a flow chart describing the function of a search manager according to an exemplary embodiment.

[0018] **FIG. 6** is a flow chart describing the function of a search conductor according to an exemplary embodiment.

[0019] **FIG. 7** is a flow chart describing the function of a partitioner according to an exemplary embodiment.

[0020] **FIG. 8** is a flow chart describing a process of setting up a partition in a search conductor according to an exemplary embodiment.

[0021] **FIG. 9A** shows a collection, its updated version, and their associated partitions according to an exemplary embodiment.

[0022] **FIG. 9B** shows a first and second search node including a first collection connected to a search manager according to an exemplary embodiment.

[0023] **FIG. 9C** shows a first search node including a first collection disconnected from a search manager and a second search node including a first collection connected to a search manager according to an exemplary embodiment.

[0024] **FIG. 9D** shows a first search node loading an updated collection, and a second search node connected to a search manager according to an exemplary embodiment.

[0025] **FIG. 9E** shows a first search node including an updated collection connected to a search manager, and a second search node including a first collection disconnected from a search manager according to an exemplary embodiment.

[0026] **FIG. 9F** shows a second search node loading an updated collection, and a first search node connected to a search manager according to an exemplary embodiment.

[0027] **FIG. 9G** shows a first and second search node including an updated collection connected to a search manager according to an exemplary embodiment.

[0028] **FIG. 10** shows a cluster of search nodes including partitions for two collections according to an exemplary embodiment.

DEFINITIONS

[0029] As used here, the following terms may have the following definitions:

[0030] **"Node"** refers to a computer hardware configuration suitable for running one or more modules.

[0031] **"Cluster"** refers to a set of one or more nodes.

[0032] **"Module"** refers to a computer software component suitable for carrying out one or more defined tasks.

[0033] **"Collection"** refers to a discrete set of records.

[0034] **"Record"** refers to one or more pieces of information that may be handled as a unit.

[0035] **"Field"** refers to one data element within a record.

[0036] **"Partition"** refers to an arbitrarily delimited portion of records of a collection.

[0037] **"Schema"** refers to data describing one or more characteristics of one or more records.

[0038] **"Search Manager"**, or **"S.M."**, refers to a module configured to at least receive one or more queries and return one or more search results.

[0039] **"Analytics Agent"**, **"Analytics Module"**, **"A.A."**, or **"A.M."**, refers to a module configured to at least receive one or more records, process said one or more records, and return the resulting one or more processed records.

[0040] **"Search Conductor"**, or **"S.C."**, refers to a module configured to at least run one or more queries on a partition and return the search results to one or more search managers.

[0041] **"Node Manager"**, or **"N.M."**, refers to a module configured to at least perform one or more commands on a node and communicate with one or more supervisors.

[0042] **"Supervisor"** refers to a module configured to at least communicate with one or more components of a system and determine one or more statuses.

[0043] **"Heartbeat"**, or **"HB"**, refers to a signal communicating at least one or more statuses to one or more supervisors.

[0044] **"Partitioner"** refers to a module configured to at least divide one or more collections into one or more partitions.

[0045] **"Dependency Manager"**, or **"D.M."**, refers to a module configured to at least include one or more dependency trees associated with one or more modules, partitions, or suitable combinations, in a system; to at least receive a request for information relating to any one or more suitable portions of said one or more dependency trees; and to at least return one or more configurations derived from said portions.

[0046] **"Database"** refers to any system including any combination of clusters and modules suitable for storing one or more collections and suitable to process one or more queries.

[0047] **"Query"** refers to a request to retrieve information from one or more suitable partitions or databases.

[0048] **"Memory"** refers to any hardware component suitable for storing information and retrieving said information at a sufficiently high speed.

[0049] **"Fragment"** refers to separating records into smaller records until a desired level of granularity is achieved.

DETAILED DESCRIPTION

[0050] Reference will now be made in detail to several preferred embodiments, examples of which are illustrated in the accompanying drawings. The embodiments described herein are intended to be exemplary. One skilled in the art recognizes that numerous alternative components and embodiments may be substituted for the particular examples described herein and still fall within the scope of the invention.

[0051] Exemplary embodiments describe an in-memory database including one or more clusters and one or more modules, where suitable modules may include one or more of a search manager, an analytics agent, a node manager, a search conductor, a supervisor, a dependency manager, and/or a partitioner.

[0052] SYSTEM CONFIGURATION

[0053] In-Memory Database Architecture

[0054] An in-memory database is a database storing data in records controlled by a database management system (DBMS) configured to store data records in a device's main memory, as opposed to conventional databases and DBMS modules that store data in "disk" memory. Conventional disk storage requires processors (CPUs) to execute read and write commands to a device's hard disk, thus requiring CPUs to execute instructions to locate (i.e., seek) and retrieve the memory location for the data, before performing some type of operation with the data at that memory location. In-memory database systems access data that is placed into main memory, and then addressed accordingly, thereby mitigating the number of instructions performed by the CPUs and eliminating the seek time associated with CPUs seeking data on hard disk.

[0055] In-memory databases may be implemented in a distributed computing architecture, which may be a computing system comprising one or more nodes configured to aggregate the nodes' respective resources (e.g., memory, disks, processors). As disclosed herein, embodiments of a computing system hosting an in-memory database may distribute and store data records of the database among one or more nodes. In some embodiments, these nodes are formed into "clusters" of nodes. In some embodiments, these clusters of nodes store portions, or "collections," of database information.

[0056] In one or more embodiments, a system interface may feed one or more search queries to one or more search managers. The search managers may be linked to one or more analytics agents that perform certain analytic techniques depending upon the embodiment and return the results to a search manager. The search managers may be linked to one or more search conductors. The search conductors may service search queries and database updates

to one or more data partitions. In one or more embodiments, one or more nodes comprising a partitioner store one or more partitions of one or more database collections. A partition of a collection stores one or more records of the collection that have been partitioned into the particular partition. Thus, the one or more nodes storing each of the partitions of a collection are storing records of the in-memory database.

[0057] Partitioner Compression

[0058] An in-memory database may be an organized collection of information stored as "records" having "fields" of information. For example, a restaurant database may have a record for each restaurant in a region, and each record contains a different field for describing each of the characteristics of the restaurant, such as name, address, type of cuisine, and the like.

[0059] An embodiment of an in-memory database may use clusters of one or more nodes to store and access data; larger amounts of data may require larger amounts of non-transitory, machine-readable storage space. Compression reduces the amount of storage space required to host the information.

[0060] In some embodiments, one or more collections may be described using any suitable schema that defines the compression technique used for one or more fields of the one or more records of the one or more collections. In these embodiments, one or more fields may be compressed by a partitioner using one or more techniques suitable for compressing the type of data stored in a field.

[0061] In some embodiments, the type of data stored in a field may be compressed after fragmentation in which records in a collection are separated into smaller records until a desired data granularity is achieved. In such embodiments, fragmented record indices may be

used to identify which record the fields were fragmented from to ensure the system remains aware that the records originate from the same original record of the collection. Fragmented records may be compressed further by according to one or more fragmenting algorithms.

[0062] In some embodiments, one or more collections may be indexed and/or compressed by one or more partitioner modules, which may be associated with one or more search conductor modules of an in-memory database system. In some embodiments, one or more compression techniques facilitate data compression while allowing data to be decompressed and/or accessed at any level of the in-memory database, including the field level, the record level, or the collection level.

[0063] **System Architecture**

[0064] **FIG. 1** shows system architecture **100** having system interface **102**, first search manager **110**, nth search manager **112**, first analytics agent **120**, nth analytics agent **122**, first search conductor **130**, nth search conductor **132**, partition data **140**, partitioner **150**, first collection **160**, nth collection **162**, supervisor **170**, and dependency manager **180**.

[0065] In one or more embodiments, system interface **102** may feed one or more queries generated outside system architecture **100** to one or more search managers **110**, **112** in a first cluster including at least one node including a first search manager **110** and up to *n* nodes including an nth search manager **112**. The one or more search managers **110**, **112** in said first cluster may be linked to one or more analytics agents **120**, **122** in a second cluster including at least a first analytics agent **120** and up to nth analytics agent **122**.

[0066] Search managers **110**, **112** in the first cluster may be linked to one or more search conductors **130**, **132** in a third cluster. The third cluster may include at least a first search conductor **130** and up to an nth search conductor **132**. Each search node (i.e., node

executing search manager **110**, **112**) may include any suitable number of search conductors **130**, **132**.

[0067] Search conductors **130**, **132** in the third cluster may be linked to one or more database nodes storing partition data **140**. Partition data **140** may include one or more partitions (i.e., arbitrarily delimited portions of records partitioned from a discrete set of records) generated by a node executing one or more partitioners **150**, which may be a module configured to at least divide one or more collections into one or more partitions. Each of the partitions may correspond to at least a first collection **160** and up to nth collection **162**. The collections **160**, **162** may additionally be described by one or more schemata files, which may define the data in the collections **160**, **162**. The one or more schemata may include information about the name of the fields in records of the partitions, whether said fields are indexed, what compression method was used, and what scoring algorithm is the default for the fields, amongst others. The schemata may be used by partitioners **150** when partitioning the first collection **160** and up to nth collection **162**, and may be additionally be used by the first search manager **110** and up nth search manager **112** when executing one or more queries on the collections.

[0068] One or more nodes may execute a supervisor **170** software module that receives a heartbeat signal transmitted from other nodes of the system **100**. A supervisor **170** may be configured to receive data from nodes of the system **100** that execute one or more dependency manager **180** software modules. A dependency manager **180** node may store, update, and reference dependency trees associated with one or more modules, partitions, or suitable combinations thereof, which may indicate configuration dependencies for nodes, modules, and partitions, based on relative relationships. A supervisor **170** may additionally be linked to other nodes in the system **100** executing one or more other supervisors **170**. In

some cases, links to additional supervisors **170** may cross between clusters of the system architecture **100**.

[0069] Nodes executing an analytics agent **120**,**122** may execute one or more suitable analytics modules, which conform to a specified application programming interface (API) that facilitates interoperability and data transfer between the components of the system (e.g., software modules, nodes). Analytics agents **120**, **122** may be configured to process aggregated query results returned from search conductors **130**, **132**. For example, a search manager **110** may receive a search query and then generate search conductor queries, which the search manager 110 issues to one or more search conductors **130**, **132**. After the search conductors **130**, **132** execute their respectively assigned search conductor queries, the search manager **110** will receive a set of aggregated query results from the one or more search conductors **130**, **132**. The search manager **110** may forward these search query results to an analytics agent **120** for further processing, if further processing is required by the parameters of the search query.

[0070] In some implementations, after a search manager **110** determines the search query has requested for an analytics agent **120** to process one or more sets of aggregated results received from the search conductors **130**, **132**, the search manager **110** may transmit a database schema file and/or one or more analytical parameters to the analytics agents **120**, **122**. In some cases, the search query may request particular analytics algorithms to be performed, which the search manager **110** may use to identify which analytics agent **120** should receive aggregated search results. In some cases, one or more of the sets of aggregated results may be transmitted to the analytics agents **120**, **122** in the form of compressed records, which contain data compressed according to a compression algorithm.

In some cases, data of the records may be compressed at the fields of the records; and in some cases, full records may be compressed.

[0071] Nodes executing analytics agents **120**, **122** having various analytics modules. Non-limiting examples may include: disambiguation modules, linking modules, and link on-the-fly modules, among other suitable modules and algorithms. As detailed later, linking modules and link-on-the-fly modules may identify, generate, and/or store metadata that links data previously stored in records of the database. Suitable modules may include any software implementation of analytical methods for processing any kind of data. In some embodiments, particular analytics modules or analytics agents **120**, **122** may be accessible only to predetermined instances, clusters, partitions, or/or instantiated objects of an in-memory database.

[0072] **Analytics Modules**

[0073] According to an embodiment, an application programming interface (API) may be used to create a plurality of analytics modules, and the disclosed system architecture may allow the addition of multiple customized analytics modules executed by analytics agents of the system, which may be added to the system architecture, without interrupting operation or services, which may support dynamic processing of constant streams of data.

[0074] Newly created analytics modules may be easily plugged into the database using simple module set up processes and may enable the application in real time to apply one or more analytical methods to aggregated results lists, without having to change how the data is managed, prepared and stored. Separate APIs may be constructed to support models which score records against queries, typically a search conductor function, or to perform closure or other aggregate analytical function on a record set, typically an analytics agent task.

[0075] **FIG. 2** is a diagram showing a configuration of a node **200**, according to an exemplary embodiment. The node **200** in **FIG. 2** may comprise a processor executing a node manager **202** software module and any number of additional software modules **210**, **212**, which may include a first software module **210** and up to nth module **212**.

[0076] According to the exemplary configuration of **FIG. 2**, the node **200** may be communicatively coupled over a data network to a second node executing a supervisor module, or supervisor node. A node manager **202** be installed and executed by the node **200** may also configured to communicate with the supervisor node, and may also be configured to monitor a software modules **210**, **212** installed on the node, including a first module **210**, up to nth module **212**. Node manager **202** may execute any suitable commands received from the supervisor, and may additionally report on the status of one or more of the node **200**, node manager **202**, and from the first module **210** to the nth module **212**. The first module **210** may be linked to the one or more supervisors and may be linked to one or more other modules in the node, where other modules in the node may be of a type differing from that of first module **210** or may share a type with first module **210**. Additionally, first module **210** may be linked with one or more other modules, nodes, or clusters in the system.

[0077] **SYSTEM OPERATION**

[0078] **System Set-up**

[0079] **FIG. 3** is a flowchart depicting node set-up **300** having steps **302**, **304**, and **306**.

[0080] In step **302**, an operating system (OS) suitable for use on a node is loaded to the node. In one or more embodiments, the OS may be loaded automatically by the node's

manufacturer. In one or more other embodiments, the OS may be loaded on the node by one or more operators.

[0081] In step **304**, a node manager suitable for use with the OS loaded on the node is installed manually by one or more operators, where the installation may determine which one or more desired modules additional to node manager will be installed on the node.

[0082] In step **306**, the node manager sends a heartbeat to a supervisor, where said heartbeat may include information sufficient for the supervisor to determine that the node is ready to receive instructions to install one or more modules.

[0083] **FIG. 4** is a flow chart depicting module set-up **400** having steps **402**, **404**, **406**, **408**, **410**, **412**, and **414**.

[0084] In step **402**, the supervisor determines one or more modules are to be installed on one or more nodes, based on the needs of the data collections defined for the system. A supervisor then sends the installation preparation instruction to one or more node managers on said one or more nodes. In some embodiments, the supervisor may track the data collections (including data shards, or portions of data) and the configuration settings associated with the respective collections. The supervisor may also be aware of all available nodes and their resources (as reported by Node Managers). The supervisor may map (i.e., correlate) the system needs to available node resources to determine which data shards or portions, and which system services or resources, should be running on each respective node. The supervisor may then sends deploy/install requests, including any dependencies defined, to the appropriate Node Managers to instruct the node managers to execute the installation on the client-side.

[0085] In step **404**, the node manager allocates the node's resources, such as computer memory, disk storage and/or a portion of CPU capacity, for running the one or more desired modules. In one or more embodiments, the allocation of resources may expire after a period of time should the supervisor discontinue the process. Non-limiting examples of resources can include computer memory, disk storage and/or a portion of CPU capacity. The resources required may be determined using the data and/or the services that the supervisor is assigning to a given node. Details of required resources may be specified in the package that defines the software and data dependencies, which is stored in the dependency manager.

[0086] In step **406**, the supervisor sends a request to a dependency manager for one or more configuration packages associated with the one or more modules to be installed on the node.

[0087] In step **408**, the supervisor may then send the configuration package to the node manager to be deployed, installed and started. The configuration package, which includes all data, software and metadata dependencies, is defined by a system administrator and stored in the dependency manager.

[0088] In step **410**, the node manager reads any software and data required to run the one or more modules from a suitable server. Suitable software and data may include software, data and metadata suitable for indexing, compressing, decompressing, scoring, slicing, joining, or otherwise processing one or more records, as well as software and data suitable for communicating, coordinating, monitoring, or otherwise interacting with one or more other components in a system.

[0089] In step **412**, the node manager installs the required software fetched in step **410**.

[0090] In step **414**, the node manager executes the software installed in step **412**.

[0091] **Query Execution**

[0092] **FIG. 5** is a flow chart depicting Query Processing **500**, having steps **502**, **504**, **508**, **510**, **512**, **514**, **518**, and **520**, and having checks **506** and **516**.

[0093] In step **502**, database queries generated by an external source, such as a browser-based graphical user interface (GUI) hosted by the system or a native GUI of the client computer, are received by one or more search managers. The queries may comprise binary data representing any suitable software source code, which may contain a user's submitted or a program's automatically-generated search parameters. The source code language used for search queries may be a data serialization language capable of handling complex data structures, such as objects or classes. Data serialization languages may be used for converting complex data objects or structures to a sequence of digital bits, and may provide a data of complex objects in a format that may be managed by most any devices. In some embodiments, the queries may be represented in a markup language, such as XML and HTML, which may be validated or otherwise understood according to a schema file (e.g., XSD). In some embodiments, queries may be represented as, or otherwise communicate, a complex data structure, such as JSON, which may be validated or otherwise understood according to a schema file. Queries may contain instructions suitable to search the database for desired records satisfying parameters of the query; and in some embodiments the suitable instructions may include a list of one or more collections to search.

[0094] In step **504**, the queries received from the external source may be parsed using according to the associated query language (e.g., SQL) by the one or more search managers, thereby generating a machine-readable query to be executed by the appropriate nodes (e.g., search conductor, analytics agent). In some cases, schema files associated with the software

language of the queries may be provided with the query, generated by code generating the query, an accepted standard, or native to the search managers. The schema files may instruct the search managers on parsing the search queries appropriately. For example, if the search queries are prepared using one or more markup languages (e.g., XML) or include a data structure (e.g., JSON), then a schema file, such as an XSD-based schema file, may be associated with the search query code or the data structure to identify and/or validate data within each of the markup tags of the XML code or the JSON code.

[0095] In check **506**, a search manager may determine, based on the user-provided or application-generated query, whether processing one or more fields of database and/or the queries should be performed. Non-limiting examples of field processing may include: address standardization, determining proximity boundaries, and synonym interpretation, among others. In some embodiments, automated or manual processes of the system may determine and identify whether any other processes associated with the search process **500** will require the use of the information included in the fields of the queries. In some embodiments, the one or more search managers may automatically determine and identify which of the one or more fields of a query may undergo a desired processing.

[0096] In step **508**, after the system determines that field processing for the one or more fields is desired in check **506**, the search managers may apply one or more suitable field processing techniques to the desired fields accordingly.

[0097] In step **510**, search managers may construct search conductor queries that are associated with the search queries. In some embodiments, the search conductor queries may be constructed so as to be processed by the various nodes of the system (e.g., search managers, search conductors, storage nodes) according to any suitable search query execution

plan, such as a stack-based search. It should be appreciated that the search queries may be encoded using any suitable binary format or other machine-readable compact format.

[0100] In step **512**, the one or more search managers send the one or more search conductor queries to one or more search conductors. In some embodiments, the search managers may automatically determine which search conductors should receive search conductor queries and then transmit the search conductor queries to an identified subset of search conductors. In such embodiments, search conductors may be pre-associated with certain collections of data; and search queries received from the system interface may specify collections to be queried. As such, the search managers transmit search conductor queries to the search conductors associated with the collections specified in the one or more search queries.

[0101] In step **514**, search conductors return search results to the corresponding search managers. In some embodiments, the search results may be returned synchronously; and in some embodiments, the search results may be returned asynchronously. Synchronously may refer to embodiments in which the search manager may block results or halt operations, while waiting for search conductor results from a particular search conductor. Asynchronously may refer to embodiments in which the search manager can receive results from many search conductors at the same time, *i.e.*, in a parallel manor, without blocking other results or halting other operations. After receiving the search results from search conductors, the search managers may collate the results received from the respective search conductors, based on record scores returned from the search conductors, into one or more results lists.

[0102] In check **516**, a search manager may determine whether additional analytics processing of the search results compiled by the search managers should be performed, based

on an indication in the search query. In some cases, the indication may be included in the search query by the user. In some embodiments, the system determines if the analytics processing is desired using information included in the search query. In some embodiments, the one or more search managers may automatically determine fields should undergo a desired analytics processing. Search queries may be constructed in a software programming language capable of conveying instructions along with other data related to the search query (e.g., strings, objects). Some programming languages, such as markup languages, may use metadata tags embedded into the code to identify various types of data, such as a field indicating a Boolean value whether analytics should be performed or a more complex user-defined field indicating a specific analytics module to be executed and/or the analytics agent node hosting the specific analytics module. Some programming languages, such as javascript or PHP, may reference stored computer files containing code that identifies whether analytics should be performed, which may be a more complex user-defined field indicating the specific analytics module to be executed and/or the analytics agent node hosting the specific analytics module.

[0103] In step **518**, if the system determines in check **516** that processing is desired, one or more analytics agents apply one or more suitable processing techniques to the one or more results lists. In one or more embodiments, suitable techniques may include rolling up several records into a more complete record, performing one or more analytics on the results, and/or determining information about relationships between records, amongst others. The analytics agent may then return one or more processed results lists to the one or more search managers.

[0104] In step **520**, the one or more search managers may decompress the one or more results lists and return them to the system that initiated the query.

[0105] **FIG. 6** is a flow diagram depicting search conductor function **600**, having steps **602**, **604**, **608**, **610**, and **612** as well as check **606**.

[0106] In step **602**, a search manager sends a query to one or more search conductors.

[0107] In step **604**, a search conductor executes the query against its loaded partition, generating a candidate result set. In one or more embodiments, step **604** may include one or more index searches. In one or more embodiments, the search conductor may use information in one or more schemata to execute the query.

[0108] In check **606**, the search conductor determines, based on the specified query, whether scoring has been requested in the search conductor query. Scoring may be indicated in the search query received by the search manager.

[0109] If scoring is requested, the search conductor scores the candidate result set in step **608**. A default score threshold may be defined in the schema, or may be included in the search conductor query sent by the search manager in step **602**. In one or more embodiments, an initial scoring may be done by the search conductor at the field level with field specific scoring algorithms, of which there may be defaults which may be overridden by one or more other scoring algorithms. Scoring algorithms may be defined or otherwise identified in the search query and/or the search conductor query, and my be performed by the search conductor accordingly. The search conductor may give the record a composite score based on those individual field scores. In some embodiments, one or more aggregate scoring methods may be applied by the search conductor, which can compute scores by aggregating one or more field scores or other aggregated scores.

[0110] In step **610**, the search conductor then uses the scores to sort any remaining records in the candidate result set.

[0111] In check **612**, the search conductor returns the candidate result set to the search manager, where the number of results returned may be limited to a size requested in the query sent by the search manager in step **602**.

[0098] **Collection Partitioning And Partition Loading**

[0112] In one or more embodiments, data may be added to one or more suitable in-memory databases.

[0113] In a first embodiment, data may be loaded in bulk using one or more partitioners.

[0114] **FIG. 7** is a flow diagram depicting collection partitioning **700**, having steps **702**, **704**, **706**, **710**, and **712**, as well as perform check **708**.

[0115] In step **702**, one or more collections are fed into one or more partitioners. The collections are fed in conjunction with one or more schemas so that the one or more partitioners can understand how to manipulate the records in the one or more collections.

[0116] In step **704**, the records in the one or more collections are fragmented.

[0117] In check **708**, the system checks the schema for the given data collection and determines whether any fields in the partitions are to be indexed by the partitioner. An index may be any suitable example of a field-index, used in any known database, such as a date index or a fuzzy index (e.g., phonetic).

[0118] In step **710**, if the system determined in check **708** that the partitioner is to index any fields in the partitions, the partitioner indexes the partitions based on the index definition in the schema.

[0119] In check **712**, the system checks the schema for the given data collection and determines whether the partitions are to be compressed by the partitioner.

[0120] In step **714**, if the system determined in check **712** that the partitioner is to compress the partitions, the partitioner compressed the fields and records using the compression methods specified in the schema, which can be any technique suitable for compressing the partitions sufficiently while additionally allowing decompression at the field level.

[0121] In step **716**, the system stores the partitions suitable for distributing the partitions to one or more search conductors.

[0122] Collection partitioning **700** may create an initial load, reload or replacement of a large data collection. The partitioner may assign unique record IDs to each record in a collection and may assign a version number to the partitioned collection, and may additionally associate the required collection schema with that partition set version for use by one or more SMs and one or more SCs.

[0123] In a second embodiment, new records may be added to a collection through one or more suitable interfaces, including a suitable query interface. The query interface may support returning result sets via queries, but may also support returning the collection schema associated with a collection version. Additionally, the search interface may allow one or more users to use that collection schema to add new records to the collection by submitting them through the search interface into the search manager. The search manager may then distribute the new record to an appropriate search conductor for addition to the collection. In some embodiments, the search manager may ensure eventual-consistency across multiple copies of a given partition and may guarantee data durability to non-volatile storage to ensure data is available after a system failure.

[0124] In one or more embodiments, records may be deleted in a similar manner. The result set from a query may include an opaque, unique ID for each record. This unique ID may encode the necessary information to uniquely identify a specific record in a given version of a collection and may include one or more of the collection name, the partition set version, and the unique record ID, amongst others. With appropriate permissions, the query interface may accept requests to delete a record corresponding to the unique record ID. This record may not be physically deleted immediately, and may be marked for deletion and may no longer be included in future answer sets.

[0125] In one or more other embodiments, a new collection schema or a delete request may be submitted to the query interface to create a new collection or remove an existing collection, respectively. A new collection created this way may start out empty, where records can be added using any suitable mechanism, including the mechanism described above.

[0126] **FIG. 8** is a flow chart depicting partition loading **800**, having steps **802**, **804**, **806**, **808**, **812**, **814**, **816**, **818** and **820**, as well as perform check **810**.

[0127] In step **802**, a supervisor determines one or more partitions are to be loaded into one or more search conductors.

[0128] In step **804**, the supervisor sends a configuration request to a dependency manager, and the dependency manager returns one or more configuration packages associated with the one or more partitions to be loaded on the one or more search conductors.

[0129] In step **806**, the supervisor determines which search conductors the partitions are to be loaded on. In one or more embodiments, the supervisor determines which one or more search conductors will be used so as to provide a desired failover ability. In one or more

other embodiments, the supervisor determines which one or more search conductors will be used so as to better level out the work load perceived by one or more clusters.

[0130] In step **808**, the supervisor sends a command to one or more node managers associated with the nodes including the one or more search conductors. In one or more embodiments, the command informs the one or more node managers to await further instructions from the supervisor for loading the partition onto the one or more search conductors. In another embodiment, the command may include the one or more configuration packages associated with the one or more partitions to be loaded into the one or more search conductors. In one or more other embodiments, the command may include instructions to prepare said one or more search conductors for loading a new partition into memory.

[0131] In step **810**, the one or more node managers allocate any node resources required for loading the partition.

[0132] In check **812**, the one or more node managers determine if one or more software or data updates are required to load the one or more partitions.

[0133] In step **814**, if the one or more node managers determined one or more software or data updates are required, the one or more node managers then retrieve said one or more software or data updates from one or more nodes suitable for storing and distributing said one or more software updates. The one or more node managers then proceed to install the one or more retrieved software or data updates.

[0134] In step **816**, the one or more node managers retrieve the one or more partitions from one or more nodes suitable for storing and distributing one or more partitions. In one or more embodiments, the retrieved partitions have previously been indexed and stored and once retrieved are loaded into memory associated with the one or more search conductors. In

another embodiment, the retrieved partitions have not been indexed or compressed previous to being retrieved, and are indexed or compressed by the one or more search conductors prior to being loaded into memory associated with the one or more search conductors.

[0135] In step **818**, the one or more search conductors send heartbeats to the supervisor and the supervisor determines the one or more search conductors are ready for use in the system.

[0136] In step **820**, the supervisor informs one or more search managers the one or more search conductors are ready to receive search requests.

[0137] **FIG. 9A** shows collection **902** and an update of collection **902** denoted collection' **910**. Collection **902** may be divided into at least a first partition **904** and up to nth partition **906**, and collection' **910** may be divided into at least a first partition' **912** and up to nth partition' **914**.

[0138] **FIG. 9B** shows first search node **920** having a first set of first partition **904** and up to nth partition **906** and second search node **930** having a second set of first partition **904** and up to nth partition **906**, where both first search node **920** and second search node **930** may be connected to at least one search manager **940**. Additionally, first search node **920**, second search node **930** and search manager **940** may be connected to one or more supervisors **950**.

[0139] **FIG. 9C** shows first search node **920** having been disconnected from search manager **940** as a result of a command from supervisor **950**, while second search node **930** still maintains a connection. In one or more embodiments, this may allow search manager **940** to run searches for records in collection **902** as first search node **920** is being upgraded.

[0140] **FIG. 9D** shows first search node **920** being updated to include collection' **910**.

[0141] **FIG. 9E** shows first search node **920** having first partition' **912** and up to nth partition' **914** connected to search manager **940** as a result of a command from supervisor **950**. supervisor **950** then sends a command to disconnect second search node **930** from search manager **940**. In one or more embodiments, this may allow search manager **940** to run searches for records in collection' **910**.

[0142] **FIG. 9F** shows second search node **930** being updated to include collection' **910**.

[0143] **FIG. 9G** shows first search node **920** having a first set of first partition' **912** and up to nth partition' **914** and second search node **930** having a second set of first partition' **912** and up to nth partition' **914** connected to search manager **940**, where the connection between second search node **930** and search manager **940** may have been re-established as a result of a command from supervisor **950**. This may allow search manager **940** to run searches for records in collection' **910** in either first search node **920** or second search node **930**.

[0144] **FIG. 10** shows search node cluster **1000**, having first search node **1002**, second search node **1004**, third search node **1006**, fourth search node **1008**, first partition **1010**, second partition **1012**, third partition **1014**, and fourth partition **1016** for a first collection, and a first partition **1020**, second partition **1022**, third partition **1024**, and fourth partition **1026** for a second collection.

[0145] Search node cluster **1000** may be arranged to as to provide a desired level of partition redundancy, where one or more search nodes may be added or removed from the system accordingly. Additionally, the partitions included in the one or more search nodes may vary with time, and may be loaded or unloaded by the search node's node manager following a process similar to partition loading **800**. When updating or otherwise changing

the partitions in search node cluster **1000**, a method similar to that described in **FIGs. 9A**, **9B**, **9C**, **9D**, **9E**, **9F**, and **9G** may be used.

[0146] **Example #1** is an in-memory database system including a search manager, an analytics agent, node managers on each node, eight search nodes each having two search conductors, a supervisor, a backup supervisor, a dependency manager, a backup dependency manager, and a partitioner on a node able to store and distribute partitions (where the node includes information for two collections split into four partitions each, collection 1 and collection 2). When a search query for records in collection 1 is received by the database, the search manager sends a query to all the search conductors having the partitioner associated with collection 1. The search conductors work asynchronously to search and score each compressed record, make a list of compressed results having a score above the threshold defined in the query, sort the list of results and return the list of compressed records to the search manager. In this example, the search conductors decompress only the fields that are to be scored. The search manager receives and aggregates the list of results from each search conductor, compiles the query result, and sends it to analytics agent for further processing. The analytics agent combines records it determines are sufficiently related, and returns the processed list of results to the search manager. The search manager then returns the final results through the system interface.

[0147] **Example #2** is an in-memory database that can perform semantic queries and return linked data results on data that is not explicitly linked in the database. Data or record linking is just one example of an aggregate analytical function that may be implemented in an Analytics Agent. This example is an in-memory database with an analytics agent capable of discovering data linkages in unlinked data and performing semantic queries and returning semantic results. Unlinked data is data from disparate data sources that has no explicit key or

other explicit link to data from other data sources. In this example, a pluggable analytics module could be developed and deployed in an Analytics Agent to discover/find data linkages across disparate data sources, based on the data content itself. When a semantic search query is executed, all relevant records are retrieved via search conductors, using non-exclusionary searches, and sent to an analytics agent where record linkages are discovered, based on the specific implementation of the analytics agent module, and confidence scores assigned. These dynamically linked records can be represented using semantic markup such as RDF/XML or other semantic data representation and returned to the user. This approach to semantic search allows unlinked data to be linked in different ways for different queries using the same unlinked data.

[0148] **Example #3** is an in-memory database that can perform graph queries and return linked data results on data that is not explicitly linked or represented in graph form in the database. This example is an in-memory database with an analytics agent capable of discovering data linkages in unlinked data and performing graph queries and returning graph query results. When a graph search query is executed, all relevant records are retrieved via search conductors, using non-exclusionary searches, and sent to an analytics agent where record linkages are discovered and confidence scores assigned. These dynamically linked records can be represented in graph form such as an RDF Graph, Property Graph or other graph data representation and returned to the user. This approach to graph search allows unlinked data to be linked in different ways for different queries using the same unlinked data.

[0149] The various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this

interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

[0150] Embodiments implemented in computer software may be implemented in software, firmware, middleware, microcode, hardware description languages, or any combination thereof. A code segment or machine-executable instructions may represent a procedure, a function, a subprogram, a program, a routine, a subroutine, a module, a software package, a class, or any combination of instructions, data structures, or program statements. A code segment may be coupled to another code segment or a hardware circuit by passing and/or receiving information, data, arguments, parameters, or memory contents. Information, arguments, parameters, data, etc. may be passed, forwarded, or transmitted via any suitable means including memory sharing, message passing, token passing, network transmission, etc.

[0151] The actual software code or specialized control hardware used to implement these systems and methods is not limiting of the invention. Thus, the operation and behavior of the systems and methods were described without reference to the specific software code being understood that software and control hardware can be designed to implement the systems and methods based on the description herein.

[0152] When implemented in software, the functions may be stored as one or more instructions or code on a non-transitory computer-readable or processor-readable storage medium. The steps of a method or algorithm disclosed herein may be embodied in a

processor-executable software module which may reside on a computer-readable or processor-readable storage medium. A non-transitory computer-readable or processor-readable media includes both computer storage media and tangible storage media that facilitate transfer of a computer program from one place to another. A non-transitory processor-readable storage media may be any available media that may be accessed by a computer. By way of example, and not limitation, such non-transitory processor-readable media may comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other tangible storage medium that may be used to store desired program code in the form of instructions or data structures and that may be accessed by a computer or processor. Disk and disc, as used herein, include compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk, and blu-ray disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media. Additionally, the operations of a method or algorithm may reside as one or any combination or set of codes and/or instructions on a non-transitory processor-readable medium and/or computer-readable medium, which may be incorporated into a computer program product.

The preceding description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the present invention. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments without departing from the spirit or scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the following claims and the principles and novel features disclosed herein.

CLAIMS

What is claimed is:

1. A computing system hosting an in-memory database, the system comprising:

a partitioner node comprising a processor configured to, in response to receiving a collection of one or more records of a database, determine whether to compress the collection based on a machine-readable schema file associated with the collection, logically partition the collection into one or more partitions according to the schema file, and distribute the one or more partitions to one or more storage nodes according to the schema file;

a storage node comprising non-transitory machine-readable main memory storing a partition received from the partitioner associated with the storage node;

a search manager node comprising a processor receiving a search query from a client device of the system, and transmitting the search queries as search conductor queries to one or more search conductors in response to receive the search query from the client device, wherein the search query is a machine-readable computer file containing parameters associated with one or more records satisfying the search query;

a search conductor node associated with one or more partitioners and comprising a processor configured to, in response to receiving a search conductor query from the search manager node: query a set of one or more partitions indicated by the search conductor query, identify one or more candidate records stored in the set of queried partitions, calculate a first score for each respective candidate record using a scoring algorithm, and transmit to the search manager a set of one or more query results containing one or more candidate records satisfying a threshold value; and

an analytics agent node comprising a processor configured to automatically generate a machine-readable computer file containing a set of one or more results derived from the set of query results, responsive to identifying in the set of query results received from the search manager node.

2. The system according to claim 1, wherein the processor of the analytics agent node is further configured to transmit the set of one or more data linkages to the search manager.

3. The system according to claim 1, wherein the processor of the search manager node is further configured to execute one or more field processing algorithms in accordance with the search query.

4. The system according to claim 1, further comprising a supervisor node comprising a processor receiving one or more heartbeat signals from one or more nodes of the system and determining a status for each of the one or more nodes based on a heartbeat signal received from each respective node, wherein each of the respective heartbeat signals indicates the status of the respective node.

5. The system according to claim 4, wherein each respective node comprises a processor configured to monitor the status of the node.

6. The system according to claim 4, further comprising a dependency manager node associated with the supervisor node and comprising a processor monitoring a node configuration status of a node monitored by the supervisor using a machine-readable dependency tree file stored in a non-transitory machine-readable storage medium.

7. The system according to claim 6, wherein the status of the heartbeat signal indicates the node configuration status, and wherein the supervisor node transmits a machine-readable configuration package file responsive to the dependency manager determining the node configuration status indicates the node is misconfigured.

8. The system according to claim 1, wherein the search conductor calculates a field score for each respective candidate record of a set of one or more updated result records, wherein the first score of each respective candidate in the set of updated result records satisfies the threshold value indicated by the search query, and transmits the updated result records to the search manager node.

9. The method according to claim 7, wherein the search conductor decompresses data stored in a candidate record in the set of updated result records using a data compression algorithm, in

response to determining the data of the candidate result record is compressed according to the data compression algorithm.

10. The system according to claim 1, further comprising a node comprising a processor executing a query interface module receiving a new collection schema file associated with one or more collections, wherein at least search conductor node is configured to automatically reconfigure one or more collections associated with the search conductor according to the new schema file.

11. The system according to claim 1, wherein the partitioner assigns a unique record identifier to each of the respective records stored in the collection according to the schema file, and generates a machine-readable index file associated with each of the partitions of the collection using the unique record identifier assigned to each respective record in the collection.

12. The system according to claim 11, wherein the search manager node distributes to the search conductor node a set of one or more new records; and wherein the search conductor automatically adds each of the new records to a partition of a collection according to the schema file, responsive to receiving the set of one or more new records.

13. The system according to claim 11, wherein the one or more search managers are further configured to receive and distribute a request to delete one or more records that correspond to a set of unique record identifiers and distribute the request to at least one search conductor; and wherein the search conductor is further configured to mark for deletion each record associated with the set of unique record identifiers.

14. The system according to claim 13, wherein marking a record for deletion precludes the record from a future search results record.

15. The system according to claim 11, wherein the unique record identifier associated with each of the records comprises one or more of a unique identifier number, a collection version number, a collection name, and a partition version number.

16. The system according to claim 11, wherein the search manager node receives a set of one or more new collections comprising one or more new records, and transmits a set of new

collections to the one or more search conductor node according to the schema file, and wherein each respective search conductor node, responsive to receiving the one or more new collections, automatically populates one or more collections associated with the respective search conductor node with the set of new one or more records in accordance with the schema file.

17. The system according to claim 1, wherein a search manager receives a request to remove a collection, the search manager processor is configured to forward the collection deletion request to a search conductor, and the search conductor is further configured to remove the collection from the database.

18. The system according to claim 1, wherein the search manager asynchronously receives each of the search result records from each of the search conductors.

19. The system according to claim 1, wherein the schema describes a collection according to one or more of names of the fields, whether the fields are indexed, a compression used, and a default scoring algorithm for the fields.

20. The system according to claim 1, wherein the analytics agent is further configured to concatenate several records into a more complete record and determine information about neighboring records to the search result records.

21. The system according to claim 1, wherein the search conductor limits the size of the search result records based on the search query received from the search manager.

22. The system according to claim 1, wherein a supervisor instructs a partitioner to compress one or more records in a collection.

23. The system according to claim 1, wherein a supervisor is further configured to determine one or more new partitions to be loaded, requests a node configuration for a node from a dependency manager, wherein the supervisor instructs a node manager of the node to retrieve the node configuration from the dependency manager;

wherein the node manager is configured to allocate memory resources of the node and loads a new partition; and

wherein the search conductor associated with the new partition in accordance with the schema informs the supervisor that the partition is loaded.

24. The system according to claim 1, wherein the analytics agent node identifies in the set of query results received from the search manager node a data linkage correlating two or more records, and wherein the data linkage correlates data contained in a first record associated with data contained in a second record

25. A computer implemented method comprising:

receiving, by a search manager computer of a system hosting an in-memory database, binary data representing a search query containing parameters querying the database, wherein the system comprises one or more storage nodes comprising main memory storing one or more collections of the database, wherein each collection contains one or more records;

transmitting, by the computer, the search query to one or more search conductor nodes according to the search query, wherein the search query indicates a set of one or more collections to be queried;

transmitting, by the computer, to one or more analytics agent nodes a set of search results based on the search query responsive to receiving from the one or more search conductors the set of search results containing one or more records satisfying the search query, wherein each respective record of the set of search results is associated with a score based on a scoring algorithm in the search query; and

responsive to the computer receiving a computer file containing a set of one or more data linkages from the one or more analytics agent nodes:

updating, by the computer, the one or more records of the set of search results according to the set of one or more data linkages received from the analytics agent nodes.

26. The method according to claim 25, wherein the computer asynchronously receives a subset of search results from each respective search conductor.

27. The method according to claim 26, wherein each subset of search records received from each respective search conductor node is ranked according to the score calculated for the respective record.

28. The method according to claim 25, wherein each respective search conductor associated with the set of collections to be queried determines a set of one or more search results containing the one or more records of the search results according to the parameters of the search query.

29. A computer-implemented method comprising:

receiving, by a computer, one or more collections from a search conductor according to a machine-readable schema file, wherein each of the collections comprises a set of one or more records having one or more fields;

partitioning, by the computer, each collection according to the schema;

compressing, by the computer, the records in the partition according to the schema; and

distributing, by the computer, each of the partitions to one or more associated search conductors to include each of the partitions in each collection corresponding to the partitioner associated with the search conductor.

30. The method according to claim 29, further comprising fragmenting, by the computer, the records in each set of records according to the schema.

31. The method according to claim 29, further comprising decompressing, by the computer, the records at a level selected from the group consisting of: a field level, a record level, a partition level, a collection level, and a database level.

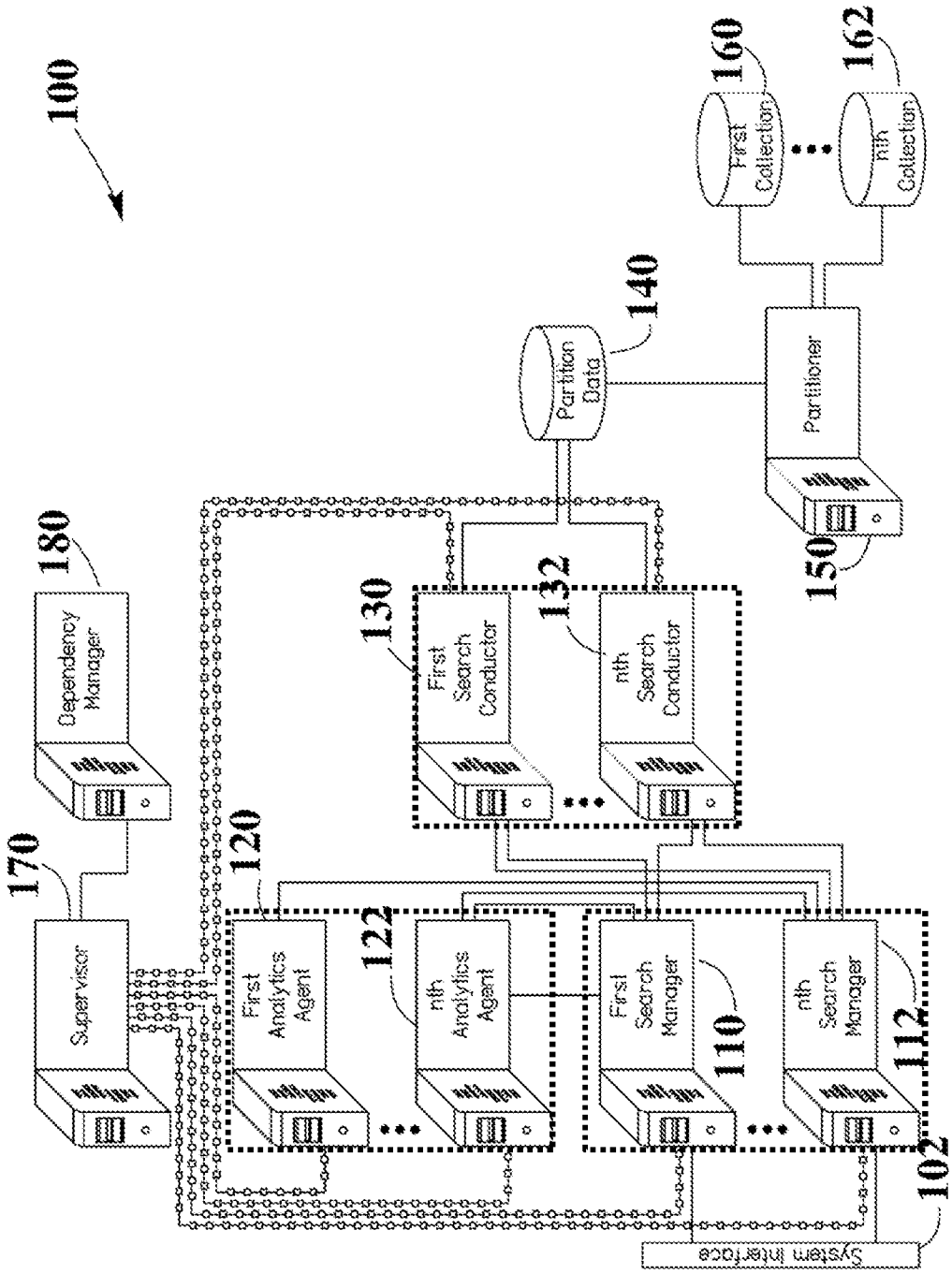
100
160
First Collection
162
nth Collection
140
Partition Data
Partitioner
150
180
Dependency Manager
130
First Search Conductor
132
nth Search Conductor
170
Supervisor
120
First Analytics Agent
122
nth Analytics Agent
First Search Manager
110
nth Search Manager
112
102
System Interface

FIG. 1

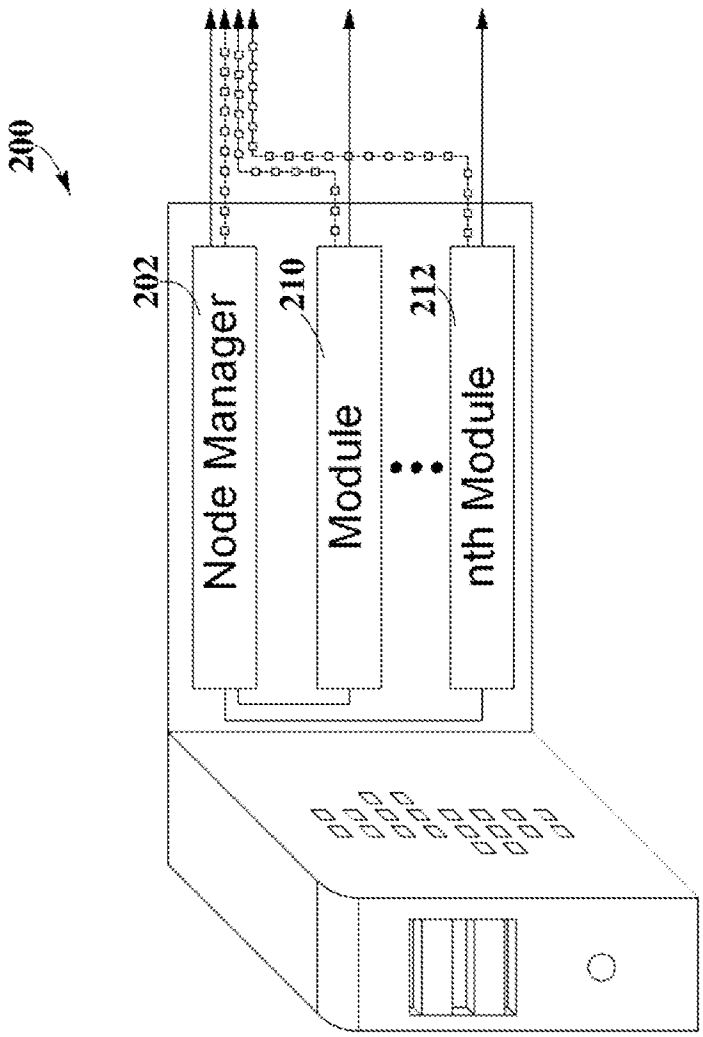

FIG. 2

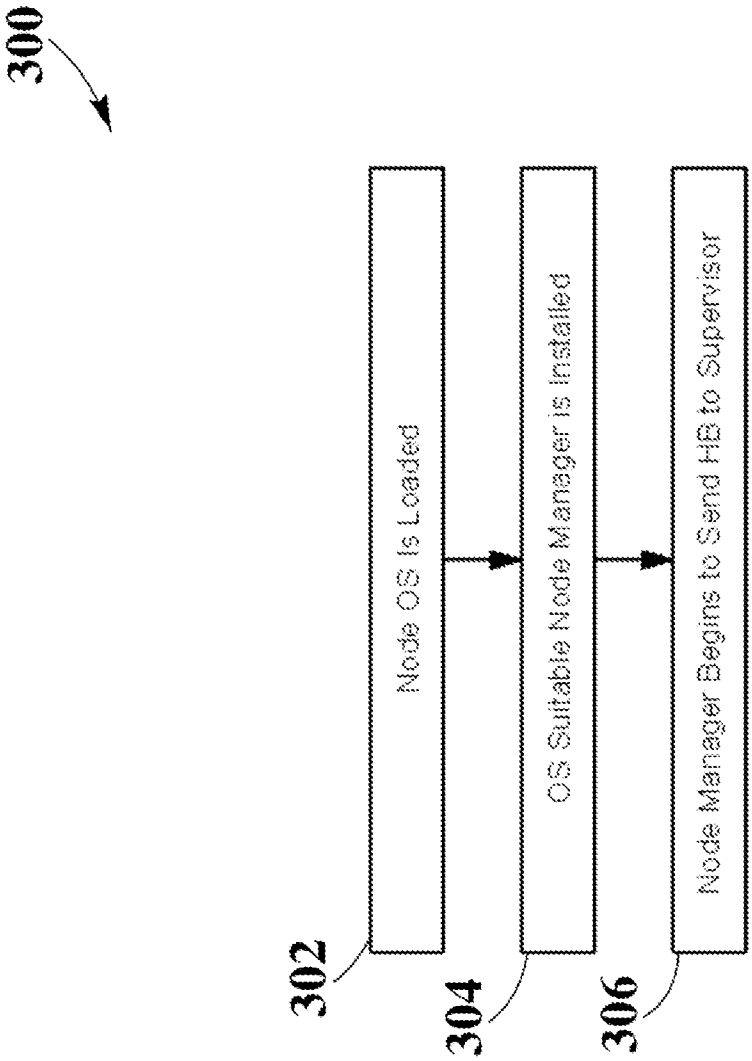
300
Node OS is Loaded
OS Suitable Node Manager is Installed
Node Manager Begins to Send HB to Supervisor
302
304
306

FIG. 3

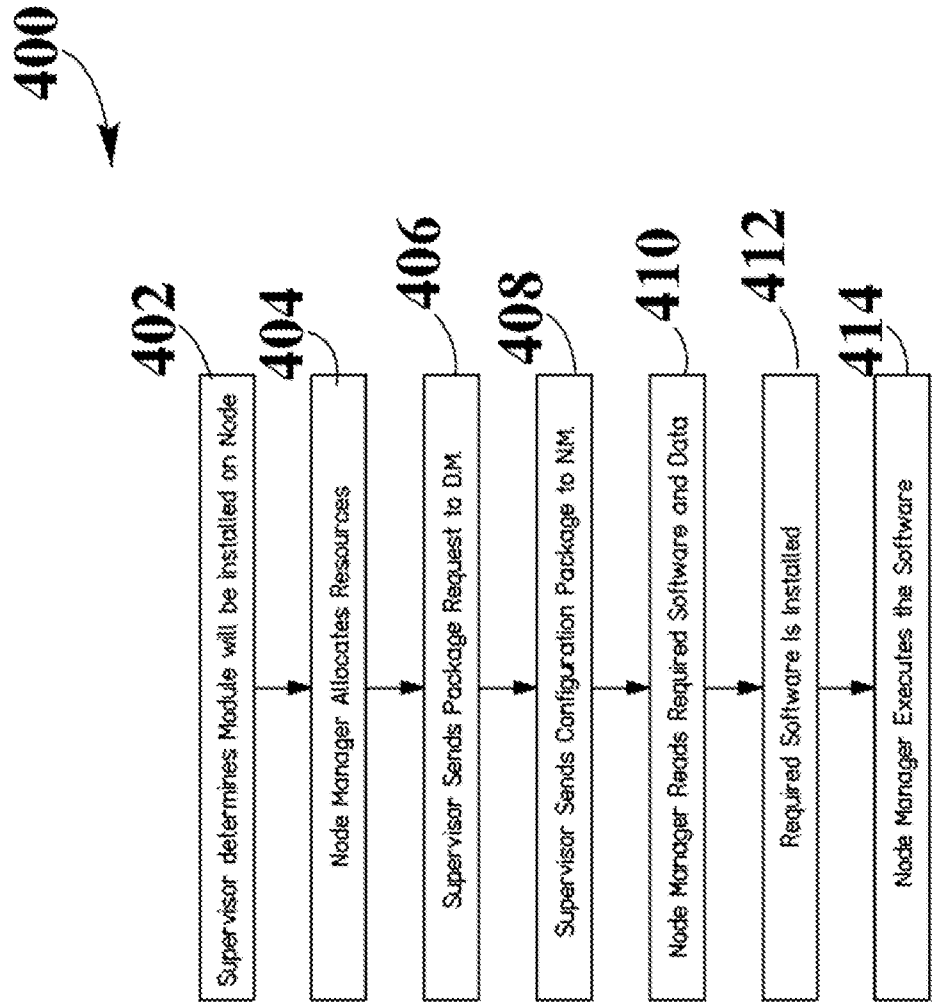
400
402
Supervisor determines Module will be installed on Node
404
Node Manager Allocates Resources
406
Supervisor Sends Package Request to D.M.
408
Supervisor Sends Configuration Package to N.M.
410
Node Manager Reads Required Software and Data
412
Required Software is installed
414
Node Manager Executes the Software

FIG. 4

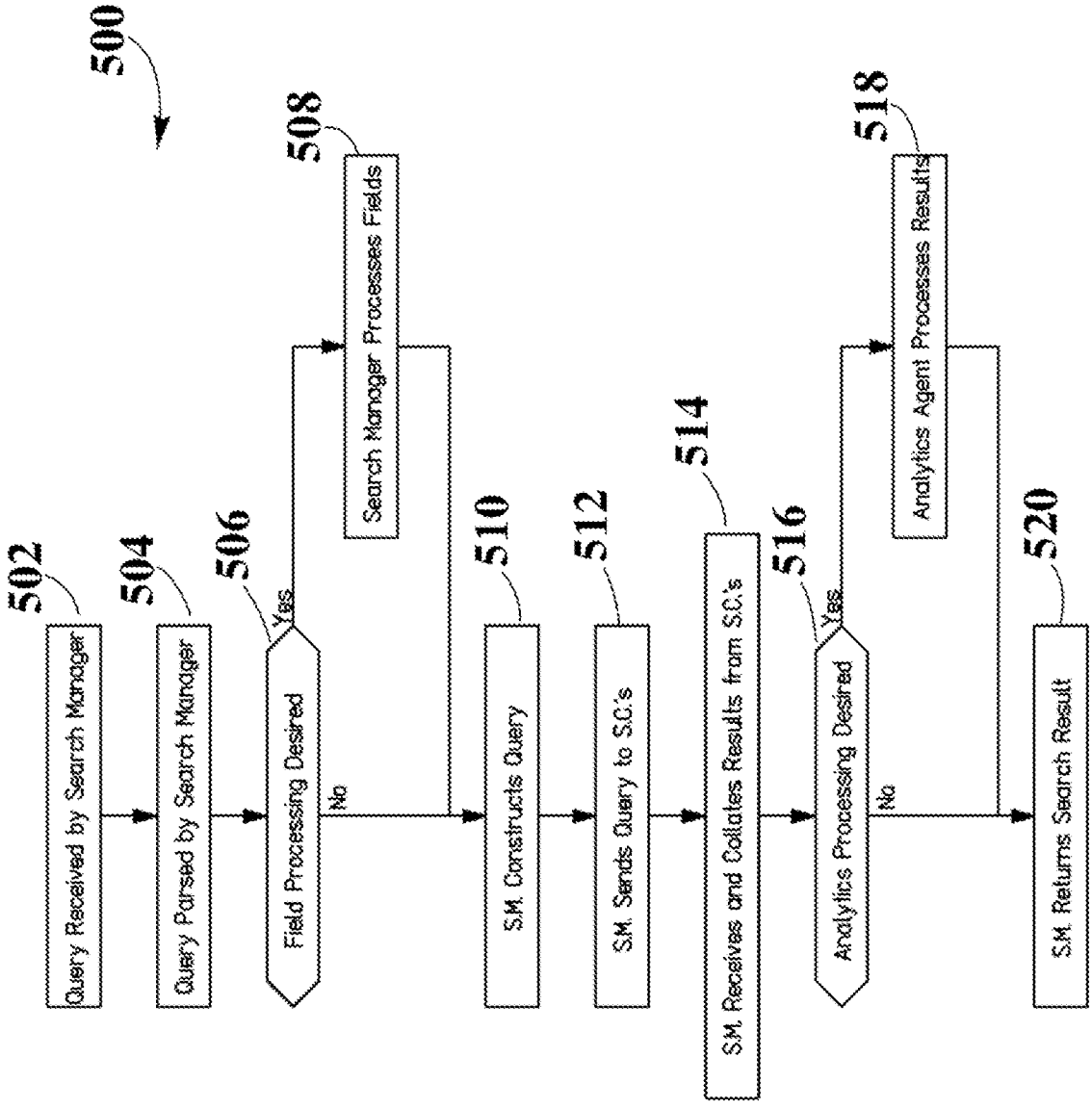
500
502
Query Received by Search Manager
504
Query Parsed by Search Manager
506
Field Processing Desired
Yes
No
508
Search Manager Processes Fields
510
S.M. Constructs Query
512
S.M. Sends Query to S.C.'s
514
S.M. Receives and Collates Results from S.C.'s
516
Analytics Processing Desired
Yes
No
518
Analytics Agent Processes Results
520
S.M. Returns Search Result

FIG. 5

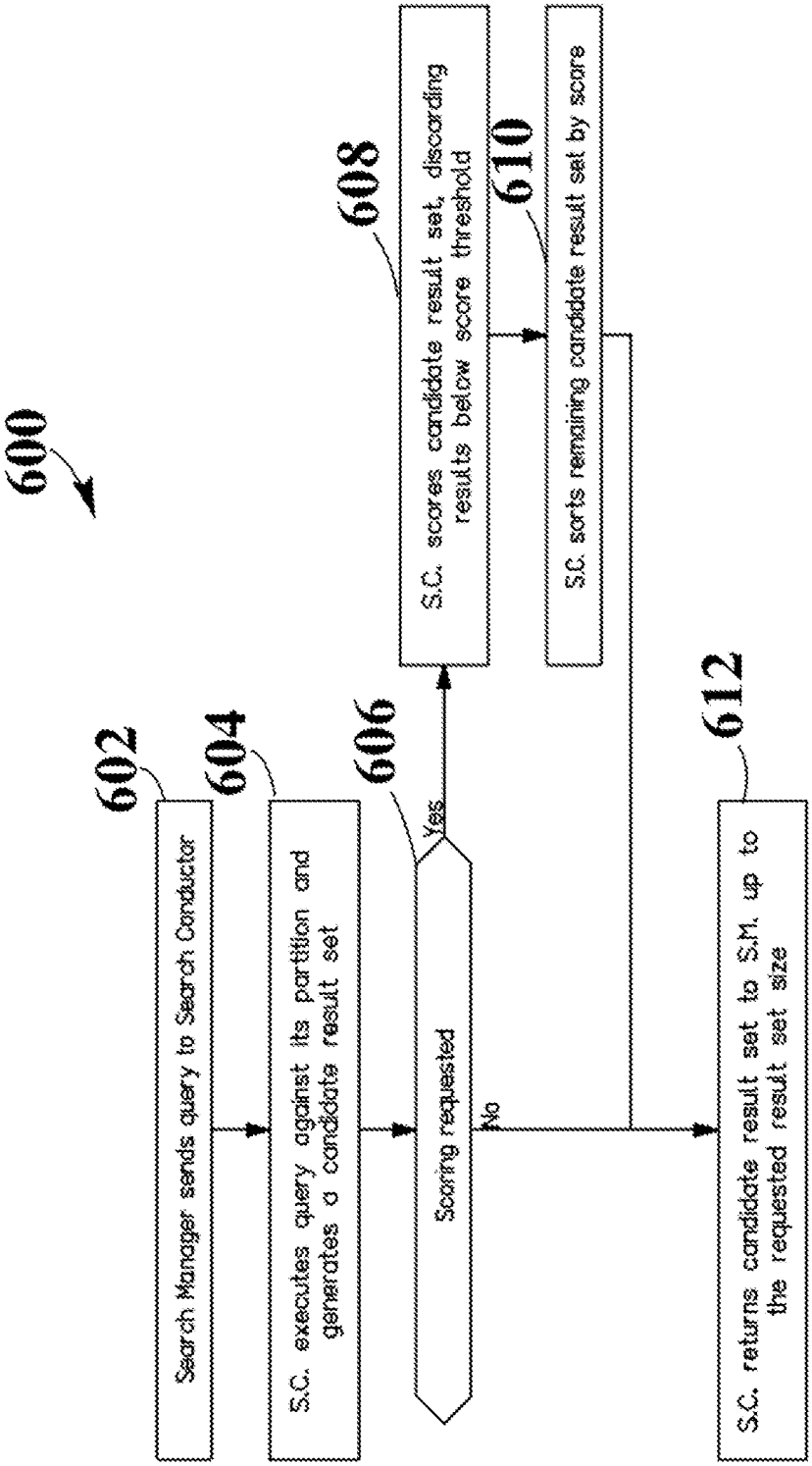
600
602
Search Manager sends query to Search Conductor
604
S.C. executes query against its partition and generates a candidate result set
606
Scoring requested
Yes
No
608
S.C. scores candidate result set, discarding results below score threshold
610
S.C. sorts remaining candidate result set by score
612
S.C. returns candidate result set to S.M. up to the requested result set size

FIG. 6

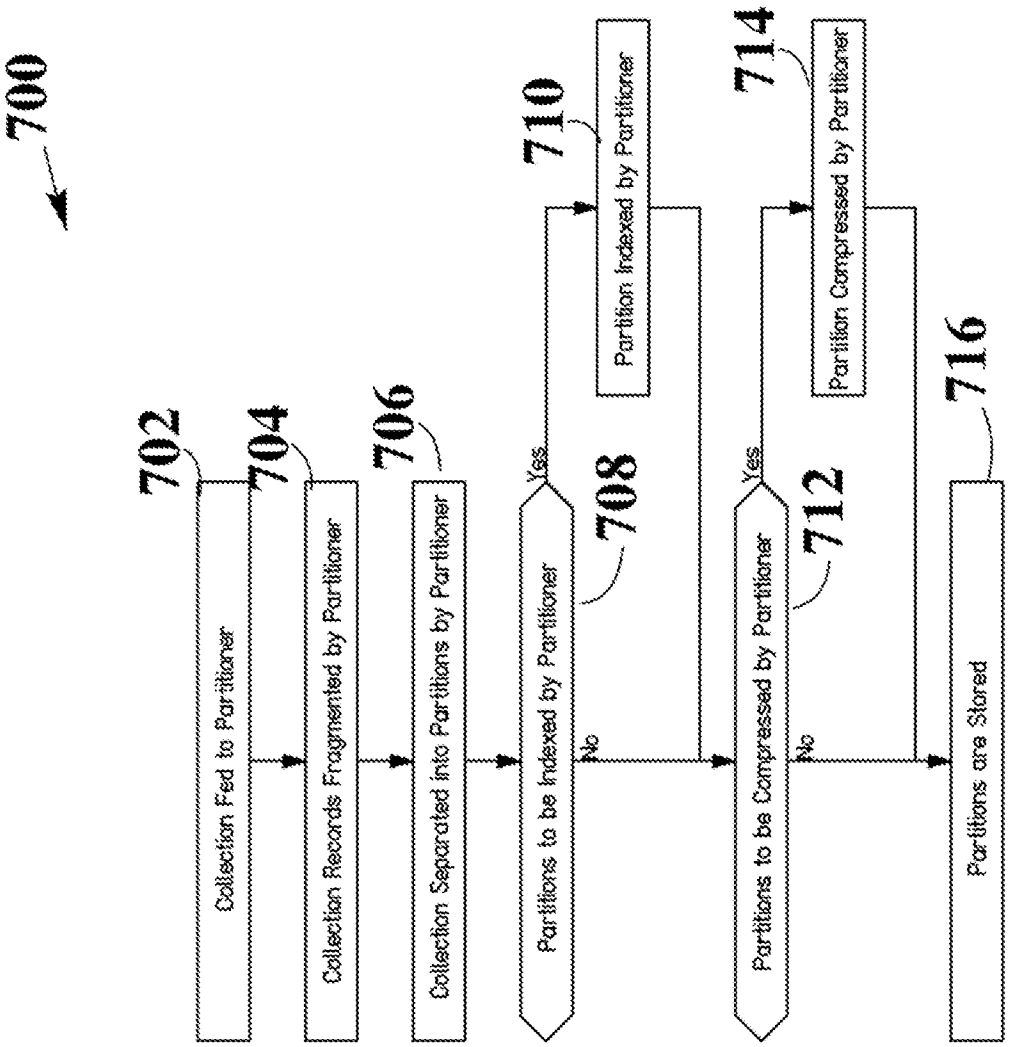
700
702
Collection Fed to Partitioner
704
Collection Records Fragmented by Partitioner
706
Collection Separated into Partitions by Partitioner
708
Partitions to be Indexed by Partitioner
Yes
No
710
Partition Indexed by Partitioner
712
Partitions to be Compressed by Partitioner
Yes
No
714
Partition Compressed by Partitioner
716
Partitions are Stored

FIG. 7

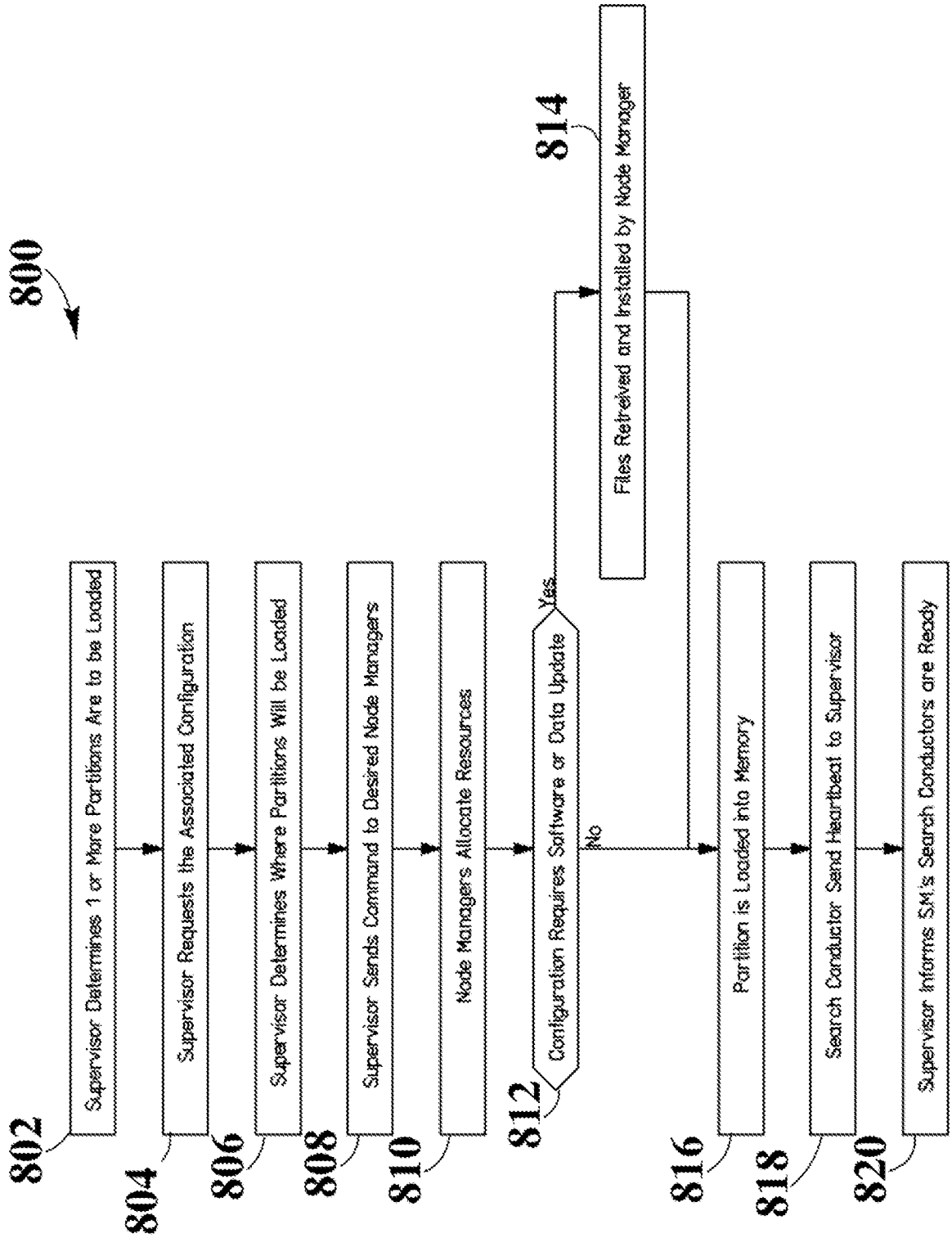
800
802
Supervisor Determines 1 or More Partitions Are to be Loaded
804
Supervisor Requests the Associated Configuration
806
Supervisor Determines Where Partitions Will be Loaded
808
Supervisor Sends Command to Desired Node Managers
810
Node Managers Allocate Resources
812
Configuration Requires Software or Data Update
Yes
814
Files Retrieved and Installed by Node Manager
No
816
Partition is Loaded into Memory
818
Search Conductor Send Heartbeat to Supervisor
820
Supervisor Informs S.M.'s Search Conductors are Ready

FIG. 8

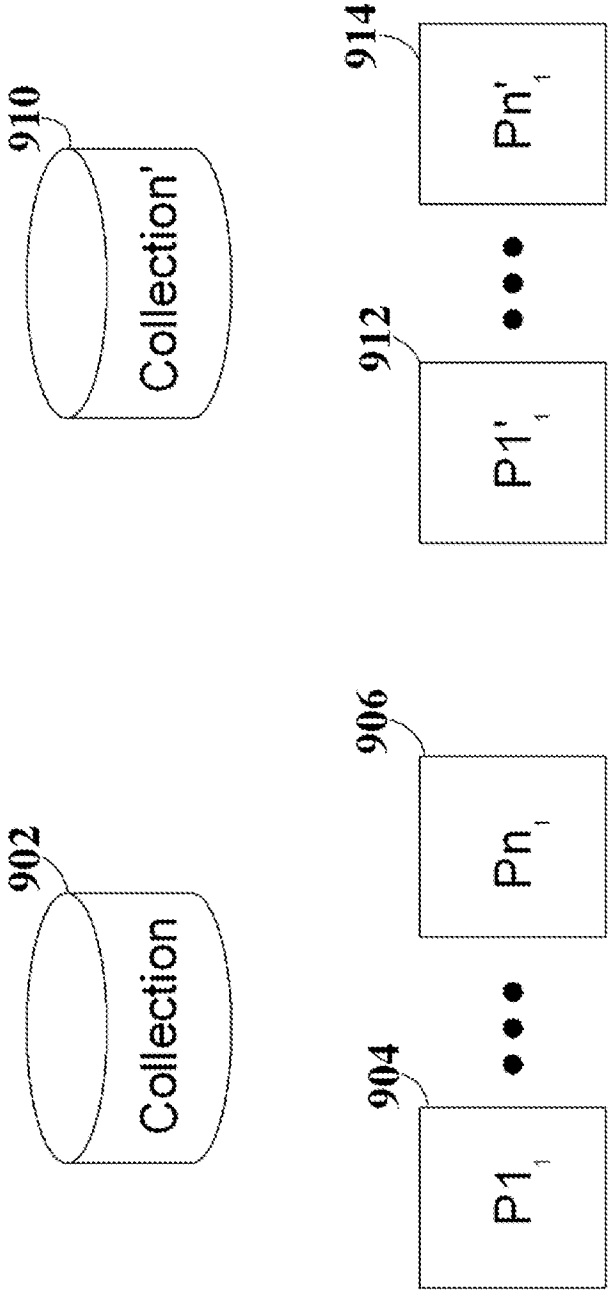

FIG. 9A

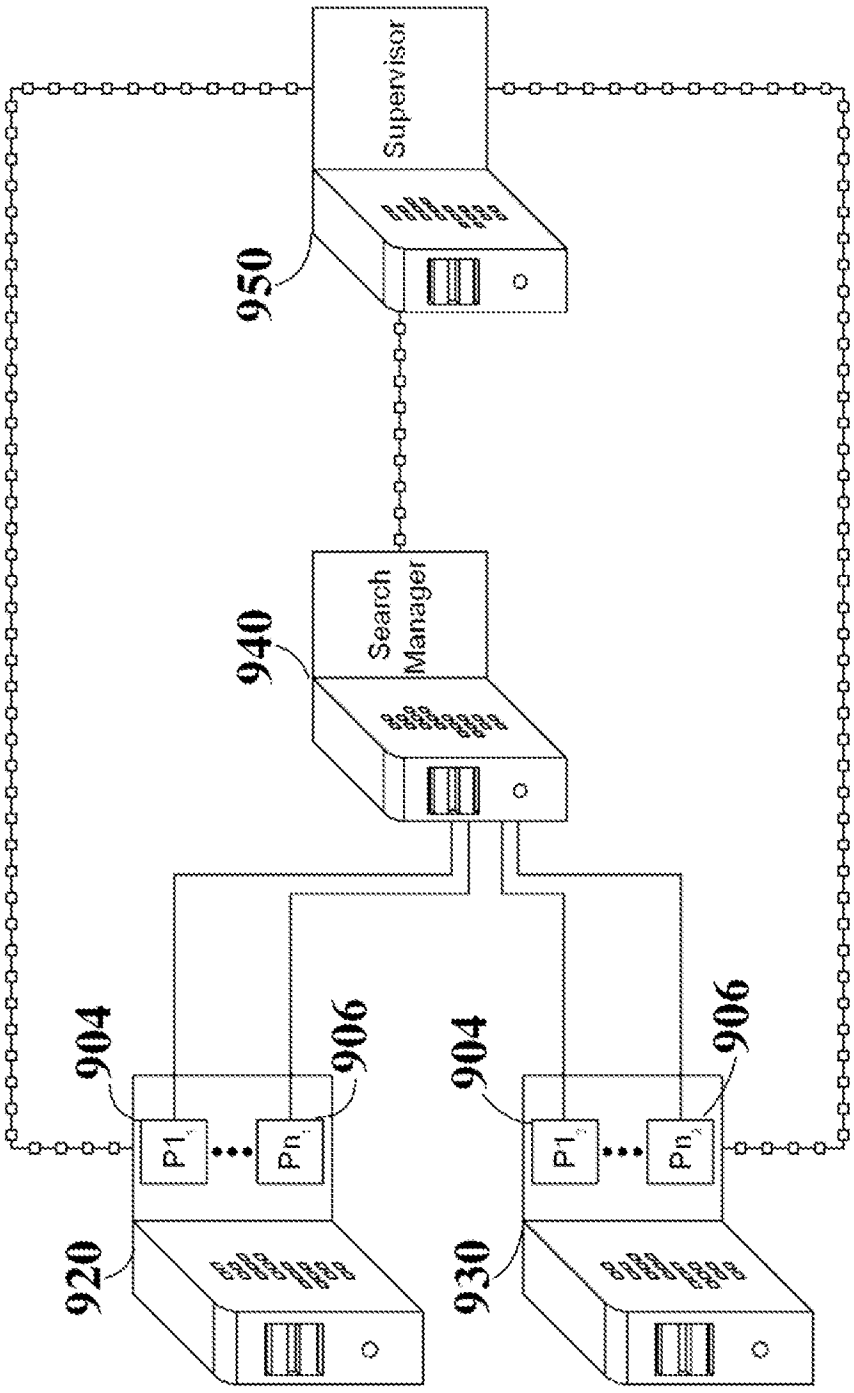

FIG. 9B

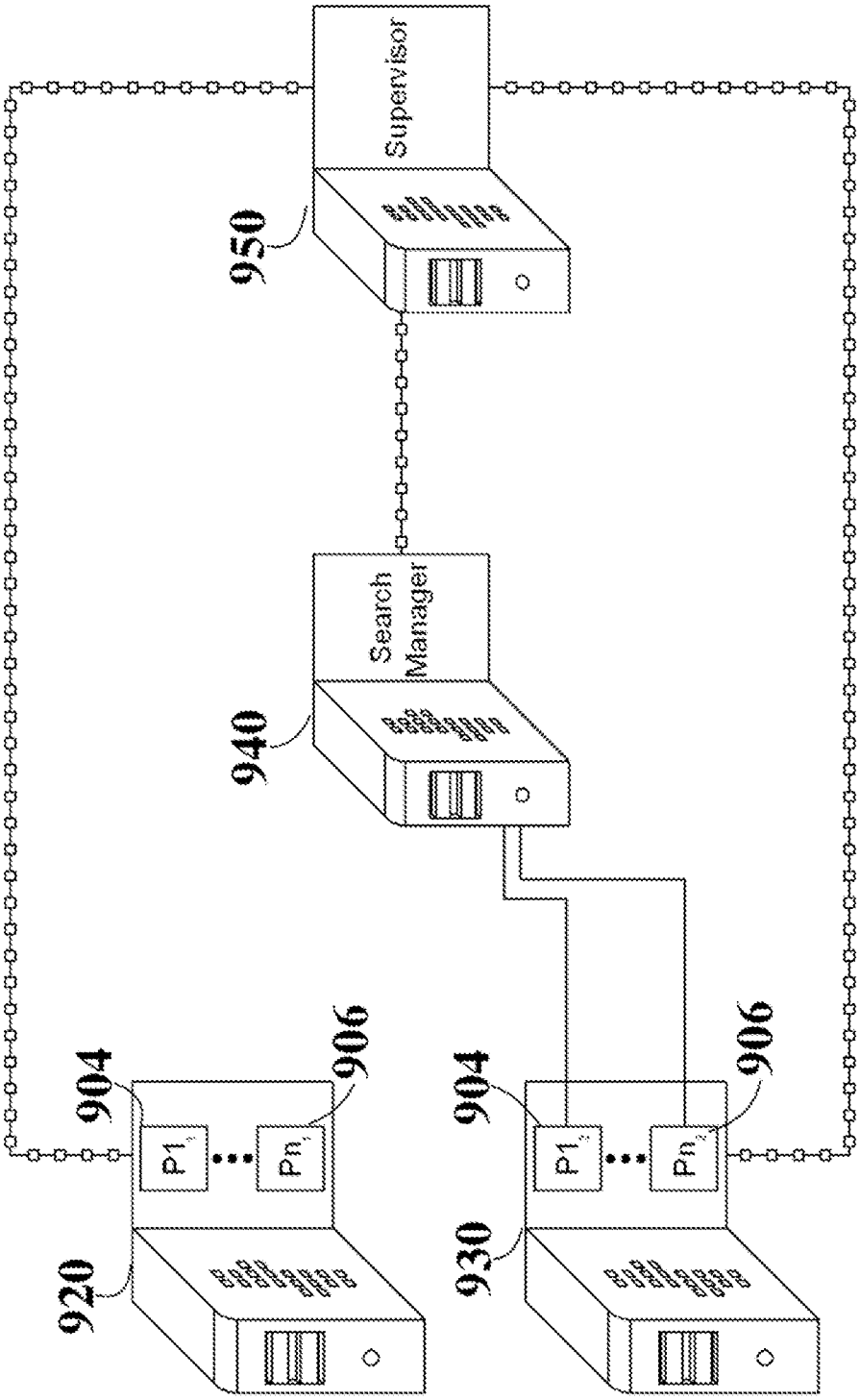
Supervisor
950
Search Manager
940
904
906
P1
Pn
920
904
906
P1
Pn
930

FIG. 9C

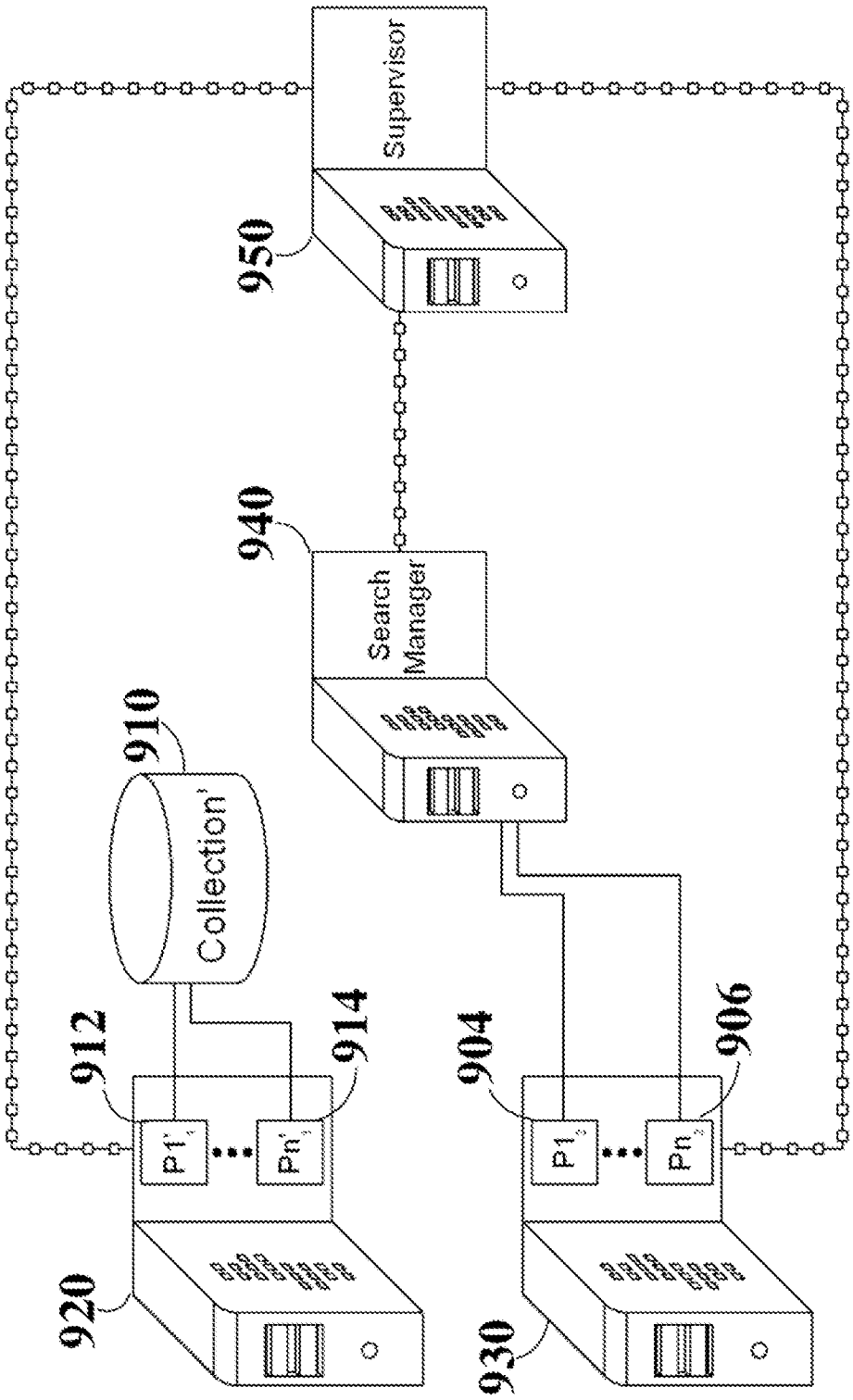
Supervisor
950
940
Search Manager
910
Collection'
912
914
904
906
920
930

FIG. 9D

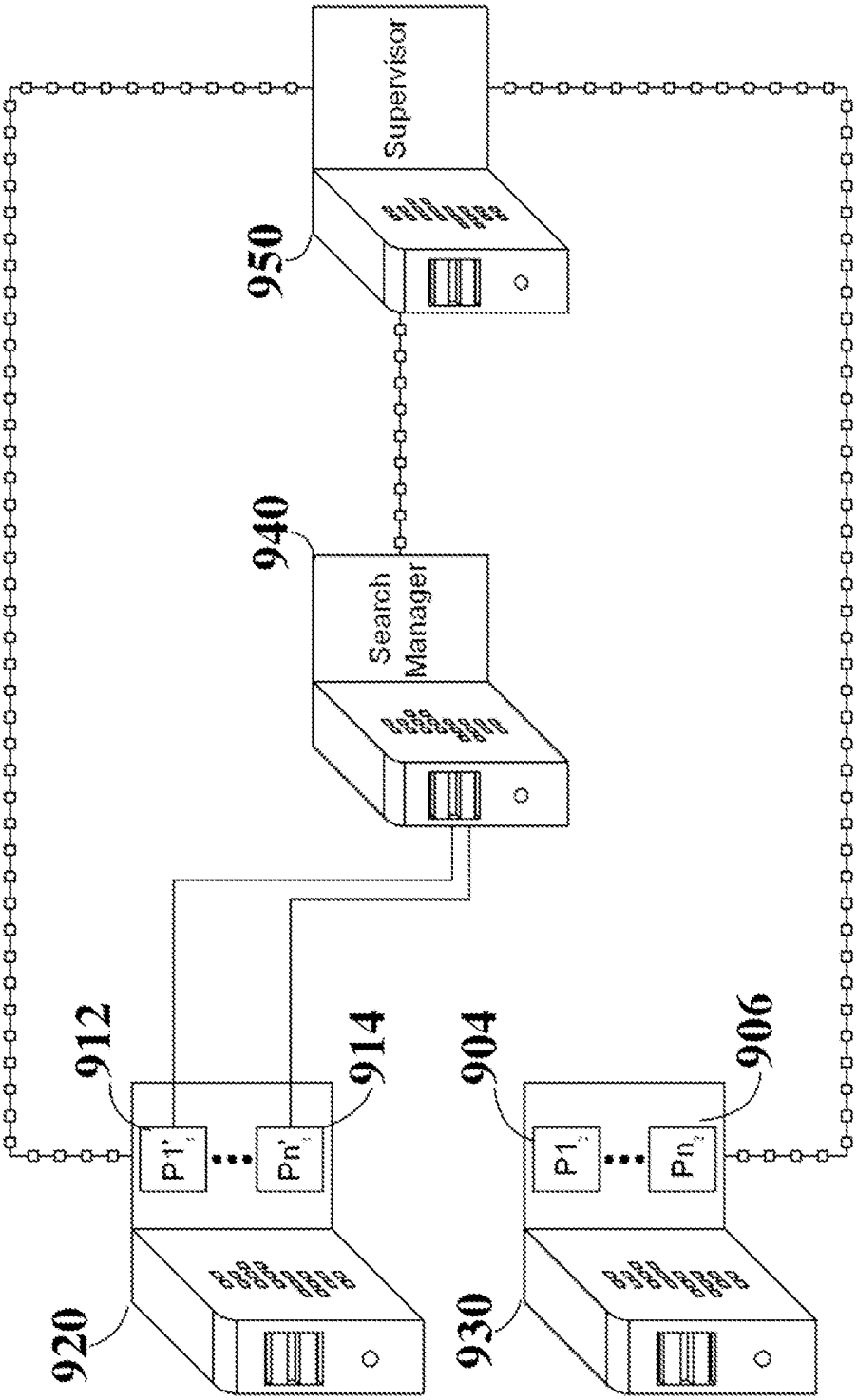
Supervisor
950
940
Search Manager
912
914
904
906
P1'
Pn'
P1''
Pn''
920
930

FIG. 9E

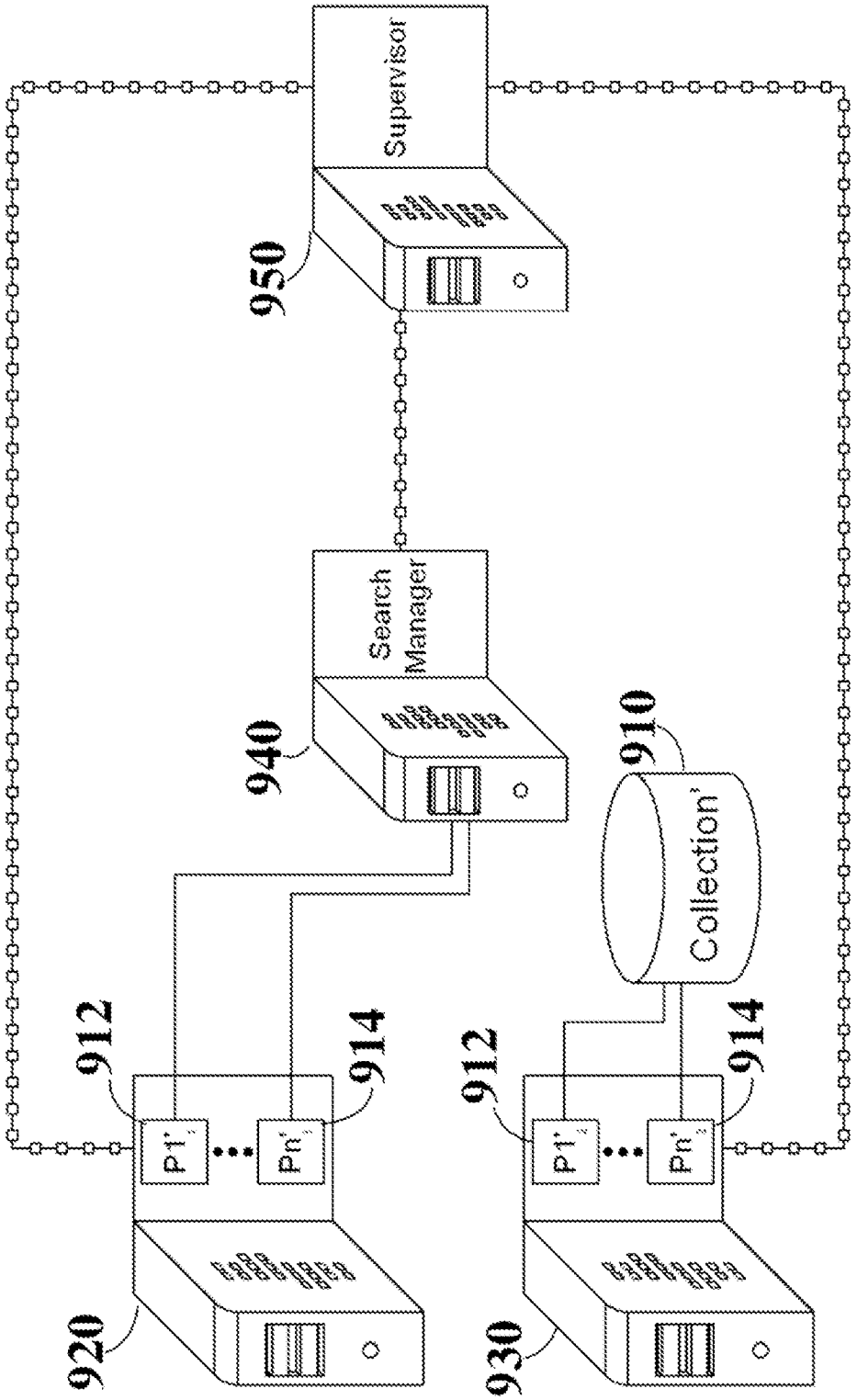
Supervisor
950
Search Manager
940
910
Collection'
912
914
P1'
Pn'
920
930

FIG. 9F

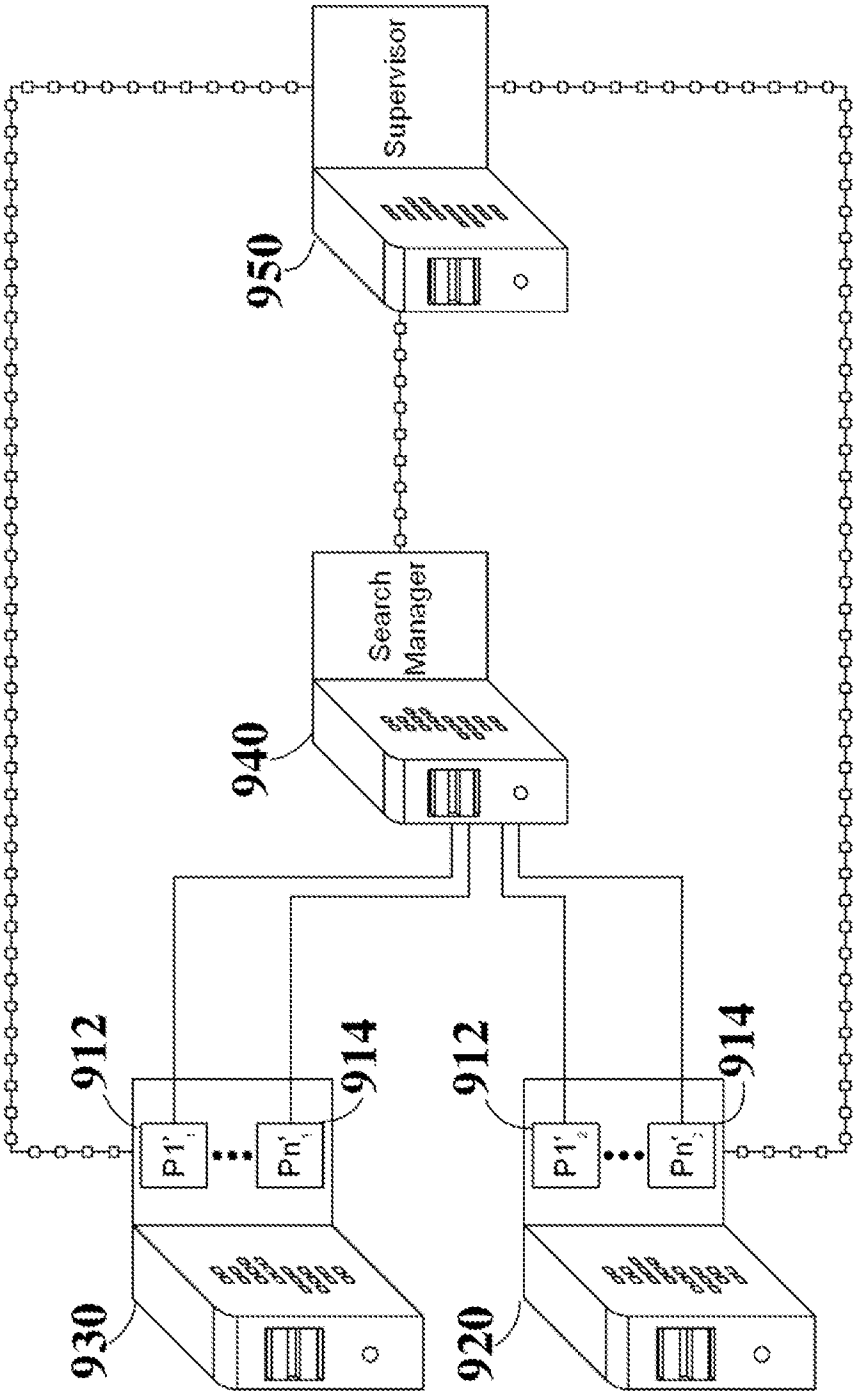
Supervisor
950
Search Manager
940
912
914
P1'1
Pn'1
930
912
914
P1'2
Pn'2
920

FIG. 9G

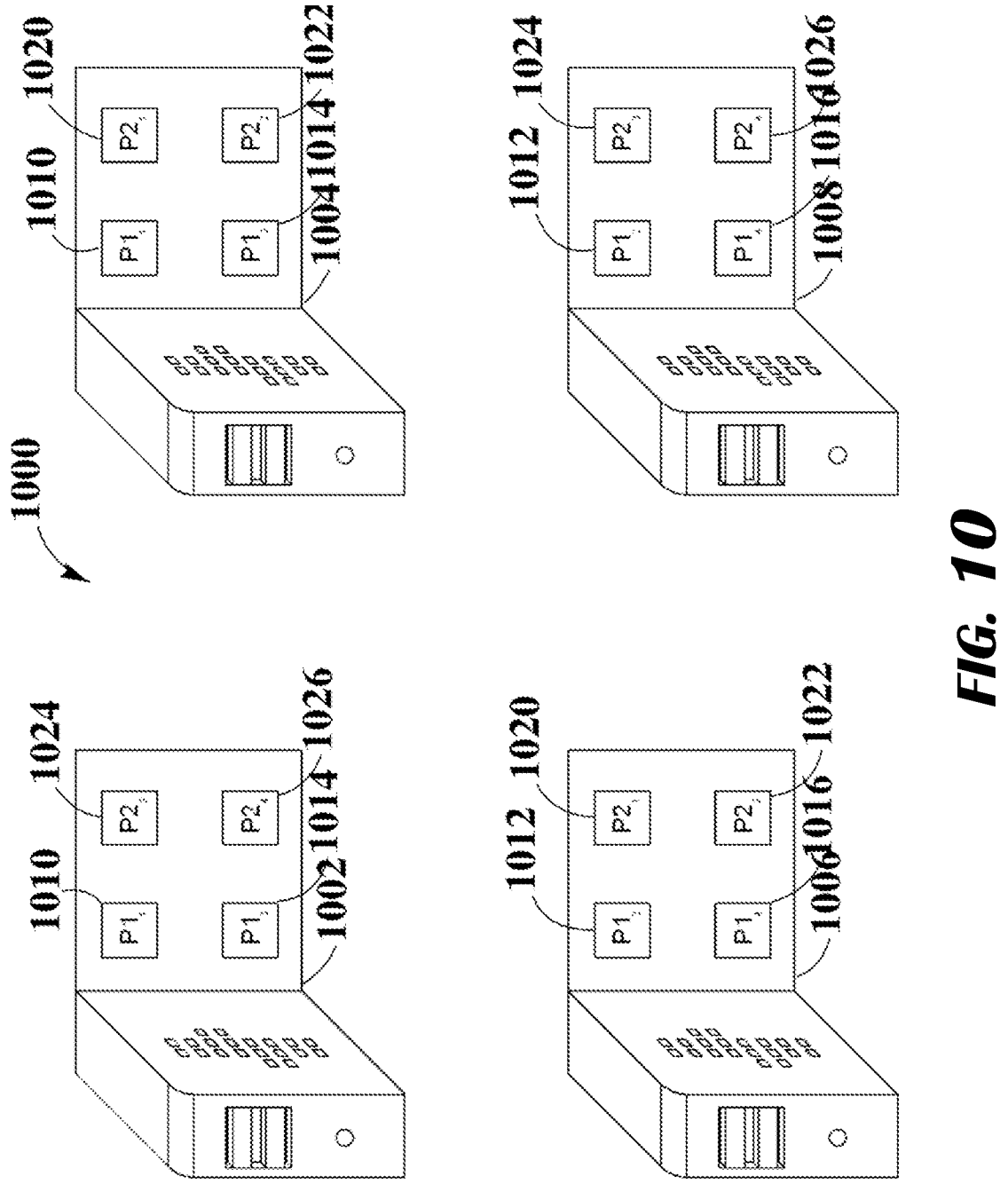
1000
1010
1020
1022
1014
1004
1012
1024
1026
1008
1002
1006
1016

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/067999

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 17/30 (2015.01)
CPC - G06F 17/30575 (2014.12)
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)
IPC(8) - G06F 17/30; G06F 7/00; G06F 12/00 (2015.01)
USPC - 707/693, E17.005, E17.014

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
CPC - G06F 17/30575; G06F 17/30545; G06F 17/30424; G06F 17/30289 (2014.12) (keyword delimited)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PatBase, Google Patents, Google Scholar.
Search terms used: in-memory, database, partition, segment, fragment, divide, compression, compaction, query, search, rank, score

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2013/0132405 A1 (BESTGEN et al.) 23 May 2013 (23.05.2013) entire document	1-31
Y	WO 2013/003770 A2 (DANI) 03 January 2013 (03.01.2013) entire document	1-24, 29-31
Y	US 2009/0322756 A1 (ROBERTSON et al) 31 December 2009 (31.12.2009) entire document	25-28
Y	US 2009/0094484 A1 (SON et al) 09 April 2009 (09.04.2009) entire document	4-7, 23
A	US 2004/0143571 A1 (BJORNSON et al) 22 July 2004 (22.07.2004) entire document	1-31
A	US 2005/0192994 A1 (CALDWELL et al) 01 September 2004 (01.09.2004) entire document	1-31
A	US 2009/0043792 A1 (BARSNESS et al) 12 February 2009 (12.02.2009) entire document	1-31

☐ Further documents are listed in the continuation of Box C. ☐

* 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
06 February 2015	10 MAR 2015
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Blaine R. Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774