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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 May 2017 (04.05.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/074649 A1

(51) International Patent Classification:
G06F 12/0813 (2016.01) *G06F 17/30* (2006.01)

(21) International Application Number:
PCT/US2016/055144

(22) International Filing Date:
3 October 2016 (03.10.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/923,111 26 October 2015 (26.10.2015) US

(71) Applicant: **MCAFEE, INC.** [US/US]; 2821 Mission College Blvd., Santa Clara, CA 95054 (US).

(72) Inventor: **LEHMANN, John**; 2821 Mission College Blvd, Santa Clara, CA 95054 (US).

(74) Agent: **KING, Jonathan, R.**; Blank Rome, LLP, 717 Texas Avenue, Suite 1400, Houston, TX 77002 (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, JP, KE, KG, 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))*

(54) Title: DYNAMIC SHARDING FOR STATE-BASED PROCESSING

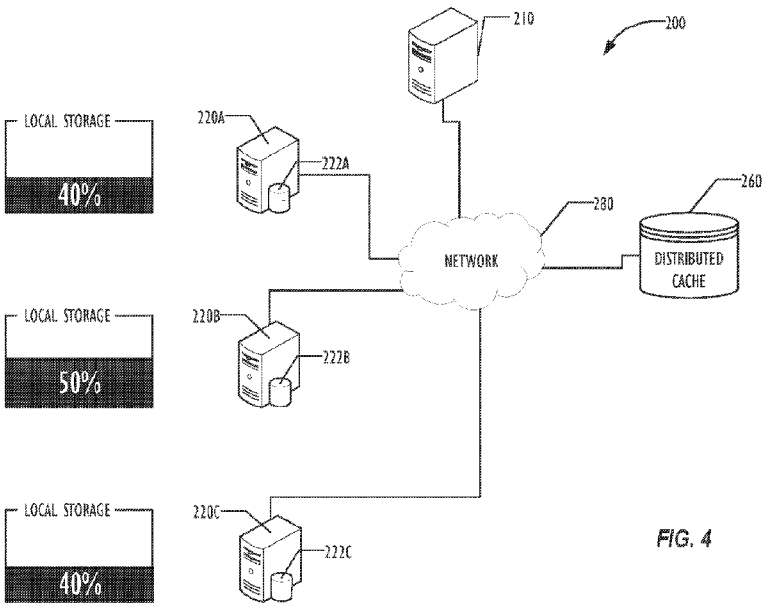


FIG. 4

(57) Abstract: Provided is a distributed data processing system to process a data set from local storage devices by dynamically real-locating portions of the data set. The system comprises a first node, a second node and a distributed cache communciatively coupled with the first node and the second node. The system is configured to transfer a portion of the data set stored in the first node to the second node via the distributed cache in response to a determination that a size of the data set stored in the first node is greater than or equal to a predetermined threshold.

WO 2017/074649 A1

Title : DYNAMIC SHARDING FOR STATE-BASED PROCESSING

DYNAMIC SHARDING FOR STATE-BASED PROCESSING

TECHNICAL FIELD

[0001] Embodiments described herein generally relate to distributed data processing and in particular to enabling a distributed data processing system to process a data set from local storage devices by dynamically reallocating portions of the data set.

BACKGROUND ART

[0002] In some data processing applications, a system can process a large amount of data by processing many smaller sets of data in sequence. Other data processing applications, however, require that the entire data set be persistently available for processing. For example, when looking at a large data set to identify correlations between two or more data points, the system must be able to access the full data set in order to identify if any data points are correlated.

[0003] While a large data set (*e.g.*, several petabytes or more) can be stored on a single storage device and processed using a single hardware element, it is preferable to process a large data set using multiple nodes of a distributed computing system so that the data can be processed more quickly. Making the large data set available to multiple nodes of a distributed computer system, however, presents its own set of challenges.

[0004] **FIG. 1** is a block diagram illustrating a conventional distributed data processing system **100** where a controller **110**, a cluster of nodes (*e.g.*, three nodes **120*a*** through **120*c***, each including a local storage device **122*a-c***), and a data store **190** communicate via a network **180**. The controller **110** outputs instructions to the nodes **120*a-c*** and the data store **190**. Each node **120** can quickly access data stored on the local data storage device **122** associated with the node **120**. For each node **120** to access data stored on the data store **190**, however, the data must first be transferred to the local data storage device **122**. Ideally, the entire data set would be divided and stored on the local storage devices **122*a-c*** so that the nodes can quickly process the data. However, because the storage capacity of each of the local storage devices **122** is finite and it is not always possible to predict how much memory will be required by each node **120**, one or more of the local storage devices **122** may run out of space.

[0005] Therefore, the conventional distributed data processing system **100** stores the majority of the data in the data store **190** and only stores a small subset of the data in the local

storage devices **122**. For example, the node **120a** may be responsible for analyzing a database with 6 billion keys with all 6 million keys (and associated data) stored in the data store **190.** The local data storage device **122*a*** may store a small subset of that database (*e.g.*, the most-recently used 100,000 keys and associated data). This process ensures that the local storage device **122** does not run out of space, but increases processing time each time the node **120*a*** must access a record that is stored in the data store **190**.

[0006] For some data processing applications, where cardinality is low, keeping a small subset of the data set in the local storage devices **122** may be sufficient. In other words, it may be possible to predict which data records to store locally with reasonable accuracy or the additional time to transfer some data records from the data store **190** may be an acceptable. In other data processing applications, however, it may be difficult or impossible to predict which data records should be stored locally.

[0007] Regardless of the predictability of the data set, partitioning and storing the entire data set on local storage devices **122** reduces processing time and eliminates the need to predict which records are best stored locally. Accordingly, there is a need for a distributed data processing system that processes a large data set from local storage devices while ensuring that the local storage devices do not run out of storage capacity.

BRIEF DESCRIPTION OF DRAWINGS

[0008] Figure **1** is a block diagram illustrating a conventional distributed data processing system.

[0009] Figure **2** is a block diagram illustrating a distributed data processing system according to an embodiment.

[0010] Figures **3-4** are block diagrams illustrating a distributed data processing system according to an embodiment.

[0011] Figure **5** is a flowchart illustrating a technique for dynamically reallocating locally-stored data in the distributed data processing system illustrated in Figures 2-4 according to one embodiment.

[0012] Figure 6 is a block diagram illustrating a computing device that may be used as node according to one embodiment.

[0013] Figure 7 is a block diagram illustrating a computing device that may be used as a node according to another embodiment.

DESCRIPTION OF EMBODIMENTS

[0014] In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the invention. It will be apparent, however, to one skilled in the art that the invention may be practiced without these specific details. In other instances, structure and devices are shown in block diagram form in order to avoid obscuring the invention. References to numbers without subscripts or suffixes are understood to reference all instance of subscripts and suffixes corresponding to the referenced number. Moreover, the language used in this disclosure has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the claims being necessary to determine such inventive subject matter. Reference in the specification to "one embodiment" or to "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least one embodiment of the invention, and multiple references to "one embodiment" or "an embodiment" should not be understood as necessarily all referring to the same embodiment.

[0015] As used herein, the term "system" refers to a single computer or a plurality of computers working together to perform the function described as being performed on or by a computer system.

[0016] As used herein, the term "node" refers to any computing device that is capable of connecting to a network. Such devices include, but are not limited to, personal computers, desktop computers, multiprocessor systems, mobile computing devices, microprocessor-based or programmable consumer electronics, network PCs, servers, network appliances, cellular phones, PDAs, or the like. Such devices may employ a wired and/or a wireless mechanism to connect to the network.

[0017] As used herein, the term "cluster" refers to a set of nodes connected by a network. Multiple nodes within a cluster may cooperate to perform the same task. The nodes within a cluster may be connected by a local area network ("LAN"), a wide area network ("WAN"), and/or the internet. Each node with a cluster may run its own instance of the same operating system. The nodes within a cluster may include the same or similar hardware.

[0018] As used herein, the term "state" refers all stored information, at a given instant in time, to which a node has access. State may be stored in random access memory (RAM) or may be persisted to non-volatile storage such as a hard drive or flash memory.

[0019] As used herein, the term "controller" refers any hardware element or software module that outputs instructions for controlling any computing device, such as a node, a cluster of nodes, and/or distributed cache.

[0020] As used herein, the term "distributed cache" refers to a storage device that stores data. The data stored in a distributed cache might be the results of an earlier computation, or the duplicates of data stored elsewhere. The distributed cache may receive data from multiple nodes via a network. A distributed cache may grow in size to store the output from the nodes, referred to by those of ordinary skill in the art as an "elastic cache." The distributed cache may include volatile random access memory (RAM) or non-volatile storage such as a hard drive or flash memory.

[0021] As used herein, the term "sharding" refers to partitioning a larger database into multiple smaller sets of data (sometimes referred to as "shards").

[0022] As used herein, the term "security information and event management (SIEM)" refers to real-time analysis of security alerts generated by network hardware and applications. SIEM may include aggregating data from multiple sources (e.g., network devices, security devices, servers, databases, applications, etc.), consolidating monitored data, correlating data/events, automating analysis of correlated data/events, outputting alerts (e.g., via email, a dashboard, etc.), identifying patterns or converting event data into charts to assist identifying patterns, gathering and/or outputting compliance data, storing historical data to facilitate correlation of data over time, retaining historical data for compliance requirements and/or for future analysis, and/or searching across logs on different nodes and time periods based on specific criteria. SIEM functions are sometimes referred to as Security information management (SIM) and/or security event management (SEM).

[0023] **FIG. 2** is a block diagram illustrating a distributed computing system **200** according to an embodiment. The system **200** includes a controller **210**, a cluster of nodes (*e.g.*, three nodes **220*a*** through **220*c*** each including a local storage device **222*a-c***), a distributed cache **260**, and a network **280**.

[0024] To mitigate the effects of a node failure, each node **220** periodically persists the data stored in the associated local storage device **222** to the distributed cache **260**. For example, the node **220*a*** may output its state from the local storage device **222*a*** to the distributed cache **260**. In response to a determination that the node **220*a*** has failed, the controller **210** may cause the previously recorded state of the node **220*a*** to be transferred from the distributed cache **260** to the storage device **222*c*** and the node **220*c*** may continue processing the data in the same manner as was previously done by node **220*a***. To mitigate the effects of a node failure, a node **220** may output information indicative of the entire data set stored on the local storage device **222** associated with the node **220** or information indicative of the changes to the data since the last time the node **220** persisted its state to the distributed cache **260**.

[0025] As described below, the system **200** also ensures that the local storage devices **222** do not run out of storage capacity using a technique of dynamically balancing the data stored on the local storage devices **222*a-c*** called "dynamic sharding."

[0026] As shown in **FIG. 3**, for example, the storage device **222*a*** may be operating at 80 percent capacity while the storage device **222*b*** is operating at 50 percent capacity. In the example shown in **FIG. 3**, the node **220*c*** is idle and, therefore, the storage device **222*c*** is operating at 0 percent capacity. The node **220*a*** may reallocate a portion of the data stored on the local storage device **222*a*** by outputting that data to the distributed cache **260** for transfer to the local storage device **222*c***. For example, the system **200** may be configured such that if a local storage devices **222*a*** reaches a predetermined threshold (*e.g.*, 80 percent), the node **220** outputs a portion of the data stored on the local storage device **222** to the distributed cache **260** for transfer to another local storage device **222**.

[0027] In the example shown in **FIGS. 3-4**, the node **220*a*** outputs half the data stored on the local storage device **222*a*** to the distributed cache **260** for transfer to the local storage device **222*c***. (In other embodiments, the node **220*a*** may output any portion of the data to any number of nodes. For example, the node **220*a*** may output two-thirds of the data stored on the local storage device **222*a*** with two nodes each receiving one-third of the data previously stored on the storage device **222*a***. The node **220*a*** may output the data for processing by one or more nodes that are currently idle or that are currently processing data but have available local storage capacity.) Accordingly, as shown in **FIG. 4**, the system ensures that all of the local storage devices **222** are below the predetermined threshold.

[0028] In order to ensure that the system **200** has the capacity to process an entire data set locally, the system **200** only requires enough nodes **220** with sufficient local storage to store the entire data set. If the data set grows, a system operator can simply add additional nodes **220** with local storage devices **222** and the system **200** will dynamically reallocate portions of the data set to the additional local storage devices **222** as needed.

[0029] The system **200** may be any distributed computing system configured to process any data. In one embodiment, the system **200** is configured to identify correlations in the data for the purposes of security information and event management (SIEM).

[0030] Partitioning the data stored in local storage device **222*a*** as illustrated in **FIG. 3** (such that it can be stored in the local storage devices **222*a*** and **222*c*** in **FIG. 4**) is referred to as sharding. The system **200** may be configured to partition the data in any suitable manner. When partitioning data used to identify correlations as described above, it is preferable that the system **200** partitions the data based on the correlation factor. For example, if the system **200** is configured to correlate one piece of data regarding a user and another piece of data regarding the same user (but is not configured to correlate the actions of multiple users), the system **200** may partition the data based on the identity of the users. Therefore, all of the data regarding each user is stored on the same local storage device **222** and is processed by the same node **220**.

[0031] **FIG. 5** is a flowchart illustrating a technique **500** for dynamically reallocating locally-stored data in a distributed data processing system **200** according to one embodiment. Data is received by the system **200** in block **510**. The data is stored in a local storage device **222*a*** in block **520**. The data stored in the local storage device **222*a*** is processed by a node **220*a*** in block **530**. The node **220*a*** periodically persists the data stored in the local storage device **222*a*** to a distributed cache **260** in block **540** so that, in the event that the node **220*a*** fails, the data can be recovered and transferred to another node **220*b-c*** for processing.

[0032] In block **550**, the node **220*a*** determines whether the amount of data stored on the local storage device **222*a*** has reached a predetermined threshold (for example, 80 percent of the total capacity of the local storage device **222*a***). If not, the node **220*a*** continues receiving, processing, and periodically backing up the data in blocks **510-540**.

[0033] If the node **220*a*** determines in block **550** that the amount of data stored on the local storage device **222*a*** has reached a predetermined threshold, the node **220*a*** outputs a portion of

the data stored on the local storage device **222*a*** to the distributed cache **260** in block **560** and outputs an instruction to the controller **210** in block **570**. In response to the instruction from the node **220*a***, the controller **210** transfers the data previously stored in the local storage device **222*a*** from distributed cache **260** to at least one other node in block **580**. For example, the data output by the node **220** may be transferred to the local storage device **222*c*** in block **580**. The additional node(s) (in this example, the node **220*c***) processes the data locally in block **590**.

[0034] The technique described above is an efficient way to reallocate data among the local storage devices **222** because it uses some of the same processes that are used to mitigate the effects of a node failure. As described above, each node **220** may already be configured to periodically persist its state to the distributed cache **260**. Similarly, in response to a determination that the amount of data stored on a local storage device **222** has reached a predetermined threshold, the node **220** associated with the local storage device **222** outputs a portion of the data to the distributed cache **260**. The technique for transferring data from the distributed cache **260** to an additional node **220** may also be similar regardless of whether a node **220** has failed or whether a local storage device **222** has reached the predetermined threshold.

[0035] Because of the similarities between the dynamic sharding technique and node failure technique, in some embodiments the system **200** may execute a single set of computer executable instructions in response to either a node failure or an indication from a node **220** that the associated local storage device **222** has reached the predetermined threshold. In particular, the controller **210** may execute many of the same process steps to transfer data from the distributed cache **260** to another node **220** regardless of whether a node has failed or whether a local storage device **222** reached a predetermined threshold. In other embodiments, the process steps for dynamic sharding and node failure may be implemented as separate computer executable instructions.

[0036] Whether the dynamic sharding and node failure techniques are implemented as a single process or separate processes, the system **200** may be configured to perform additional process steps depending in response to a determination that a local storage device **222** has reached a predetermined threshold. For example, because the data stored in the local storage device **222** may have changed since the last time the node **220** persisted its state to the distributed cache **260**, the node **220** may immediately begin the process of outputting the data

to the distributed cache **260** in response to the determination that the amount of data stored on a local storage device **222** has reached a predetermined threshold.

[0037] The local storage devices **222** may be any computer readable storage medium. In a preferred embodiment, the local storage device **222** is a high speed storage medium such as random access memory (RAM). The distributed cache **260** may include one or more computer readable storage media. In a preferred embodiment, the distributed cache includes at least one high speed storage medium, such as RAM, of sufficient size as to receive all of the data stored on one of the local storage devices **222**, and nonvolatile memory (*e.g.*, a solid state hard drive) with enough capacity to store the entire data set. In the preferred embodiment described above, when one of the local storage devices **222** persists its state to the distributed cache **260** (either to back-up the data in case of node failure or for transfer to another node), the distributed cache **260** quickly reads the data output by the node **220** into RAM and then transfers the data to nonvolatile memory. The distributed cache **260** may include a cluster of high speed storage mediums to quickly read in data output by a plurality of nodes. The distributed cache **260** may be an elastic cache that is expandable to accommodate additional nodes **220** that may be added to the system **200**.

[0038] Referring now to FIG. **6**, a block diagram illustrates a programmable device **600** that may be used for implementing the techniques described herein in accordance with one embodiment. The programmable device **600** illustrated in FIG. **6** is a multiprocessor programmable device that includes a first processing element **670** and a second processing element **680**. While two processing elements **670** and **680** are shown, an embodiment of programmable device **600** may also include only one such processing element.

[0039] Programmable device **600** is illustrated as a point-to-point interconnect system, in which the first processing element **670** and second processing element **680** are coupled via a point-to-point interconnect **650**. Any or all of the interconnects illustrated in FIG. **6** may be implemented as a multi-drop bus rather than point-to-point interconnects.

[0040] As illustrated in FIG. **6**, each of processing elements **670** and **680** may be multicore processors, including first and second processor cores (i.e., processor cores **674a** and **674b** and processor cores **684a** and **684b**). Such cores **674a**, **674b**, **684a**, **684b** may be configured to execute instruction code. However, other embodiments may use processing elements that are

single core processors as desired. In embodiments with multiple processing elements **670**, **680**, each processing element may be implemented with different numbers of cores as desired.

[0041] Each processing element **670**, **680** may include at least one shared cache **646**. The shared cache **646a**, **646b** may store data (e.g., instructions) that are utilized by one or more components of the processing element, such as the cores **674a**, **674b** and **684a**, **684b**, respectively. For example, the shared cache may locally cache data stored in a memory **632**, **634** for faster access by components of the processing elements **670**, **680**. In one or more embodiments, the shared cache **646a**, **646b** may include one or more mid-level caches, such as level 2 (L2), level 3 (L3), level 4 (L4), or other levels of cache, a last level cache (LLC), or combinations thereof.

[0042] While FIG. **6** illustrates a programmable device with two processing elements **670**, **680** for clarity of the drawing, the scope of the present invention is not so limited and any number of processing elements may be present. Alternatively, one or more of processing elements **670**, **680** may be an element other than a processor, such as an graphics processing unit (GPU), a digital signal processing (DSP) unit, a field programmable gate array, or any other programmable processing element. Processing element **680** may be heterogeneous or asymmetric to processing element **670**. There may be a variety of differences between processing elements **670**, **680** in terms of a spectrum of metrics of merit including architectural, microarchitectural, thermal, power consumption characteristics, and the like. These differences may effectively manifest themselves as asymmetry and heterogeneity amongst processing elements **670**, **680**. In some embodiments, the various processing elements **670**, **680** may reside in the same die package.

[0043] First processing element **670** may further include memory controller logic (MC) **672** and point-to-point (P–P) interconnects **676** and **678**. Similarly, second processing element **680** may include a MC **682** and P–P interconnects **686** and **688**. As illustrated in FIG. **6**, MCs **672** and **682** couple processing elements **670**, **680** to respective memories, namely a memory **632** and a memory **634**, which may be portions of main memory locally attached to the respective processors. While MC logic **672** and **682** is illustrated as integrated into processing elements **670**, **680**, in some embodiments the memory controller logic may be discrete logic outside processing elements **670**, **680** rather than integrated therein.

[0044] Processing element **670** and processing element **680** may be coupled to an I/O subsystem **690** via respective P–P interconnects **676** and **686** through links **652** and **654**. As illustrated in FIG. **6**, I/O subsystem **690** includes P–P interconnects **694** and **698**. Furthermore, I/O subsystem **690** includes an interface **692** to couple I/O subsystem **690** with a high performance graphics engine **638**. In one embodiment, a bus (not shown) may be used to couple graphics engine **638** to I/O subsystem **690**. Alternately, a point-to-point interconnect **639** may couple these components.

[0045] In turn, I/O subsystem **690** may be coupled to a first link **616** via an interface **696**. In one embodiment, first link **616** may be a Peripheral Component Interconnect (PCI) bus, or a bus such as a PCI Express bus or another I/O interconnect bus, although the scope of the present invention is not so limited.

[0046] As illustrated in FIG. **6**, various I/O devices **614**, **624** may be coupled to first link **616**, along with a bridge **618** that may couple first link **616** to a second link **620**. In one embodiment, second link **620** may be a low pin count (LPC) bus. Various devices may be coupled to second link **620** including, for example, a keyboard/mouse **612**, communication device(s) **626** (which may in turn be in communication with the computer network **603**), and a data storage unit **628** such as a disk drive or other mass storage device which may include code **630**, in one embodiment. The code **630** may include instructions for performing embodiments of one or more of the techniques described above. Further, an audio I/O **624** may be coupled to second link **620**.

[0047] Note that other embodiments are contemplated. For example, instead of the point-to-point architecture of FIG. **6**, a system may implement a multi-drop bus or another such communication topology. Although links **616** and **620** are illustrated as busses in FIG. **6**, any desired type of link may be used. In addition, the elements of FIG. **6** may alternatively be partitioned using more or fewer integrated chips than illustrated in FIG. **6**.

[0048] Referring now to FIG. **7**, a block diagram illustrates a programmable device **700** according to another embodiment. Certain aspects of FIG. **7** have been omitted from FIG. **7** in order to avoid obscuring other aspects of FIG. **7**.

[0049] FIG. **7** illustrates that processing elements **770**, **780** may include integrated memory and I/O control logic ("CL") **772** and **782**, respectively. In some embodiments, the **772**, **782** may include memory control logic (MC) such as that described above in connection with FIG.

6. In addition, CL **772**, **782** may also include I/O control logic. FIG. 7 illustrates that not only may the memories **732**, **734** be coupled to the CL **772**, **782**, but also that I/O devices **744** may also be coupled to the control logic **772**, **782**. Legacy I/O devices **715** may be coupled to the I/O subsystem **790** by interface **796**. Each processing element **770**, **780** may include multiple processor cores, illustrated in FIG. 7 as processor cores **774A**, **774B**, **784A** and **784B**. As illustrated in FIG. 7, I/O subsystem **790** includes point-to-point (P–P) interconnects **794** and **798** that connect to P-P interconnects **776** and **786** of the processing elements **770** and **780** with links **752** and **754**. Processing elements **770** and **780** may also be interconnected by link **750** and interconnects **778** and **788**, respectively.

[0050] The programmable devices depicted in FIGs. **6** and 7 are schematic illustrations of embodiments of programmable devices that may be utilized to implement various embodiments discussed herein. Various components of the programmable devices depicted in FIGs. **6** and 7 may be combined in a system-on-a-chip (SoC) architecture.

[0051] The following examples pertain to further embodiments.

[0052] Example 1 is a system, comprising: a first node that processes data stored in a first local storage device; a second node including a second local storage device; and a distributed cache in network communication with the first node and the second node, wherein the system is configured to: determine whether a size of the data stored in the first local storage device is greater than or equal to a predetermined threshold; and transfer a portion of the data stored in the first local storage device to the second local storage device via the distributed cache in response to a determination that the size of the data stored in the first local storage device is greater than or equal to the predetermined threshold.

[0053] In Example 2 the subject matter of Example 1 optionally includes wherein the predetermined threshold is a ratio of the size of the data stored in the first local storage device relative to a capacity of the first local storage device.

[0054] In Example 3 the subject matter of any of Examples 1 or 2 optionally includes wherein the first local storage device is in direct communication with the first node.

[0055] In Example 4 the subject matter of any of Examples 1 or 2 optionally includes wherein the first local storage device includes volatile memory.

[0056] In Example 5 the subject matter of any of Examples 1 or 2 optionally includes wherein the distributed cache includes nonvolatile memory.

[0057] In Example 6 the subject matter of any of Examples 1 or 2 optionally includes further comprising a controller that outputs instructions to the first node, the second node, and the distributed cache.

[0058] In Example 7 the subject matter of any of Examples 1 or 2 optionally includes wherein the first node periodically outputs information indicative of the data stored in the first local storage device to the distributed cache.

[0059] In Example 8 the subject matter of Example 7 optionally includes wherein the system is further configured to: determine whether the first node has experienced a failure; and transfer the information indicative of the data stored in the first local storage device from the distributed cache to the second local storage device in response to a determination that the first node has experienced a failure.

[0060] Example 9 is a method, comprising: storing data in a first local storage device in communication with a first node; determining whether a size of the data stored in the first local storage device is greater than or equal to a predetermined threshold; and transferring a portion of the data stored in the first local storage device to a second local storage device in direct communication with a second node via a distributed cache in response to a determination that the size of the data stored in the first local storage device is greater than or equal to the predetermined threshold.

[0061] In Example 10 the subject matter of Example 9 optionally includes wherein the predetermined threshold is a ratio of the size of the data stored in the first local storage device relative to a capacity of the first local storage device.

[0062] In Example 11 the subject matter of any of Examples 9 or 10 optionally includes wherein the first local storage device includes volatile memory.

[0063] In Example 12 the subject matter of any of Examples 9 or 10 optionally includes wherein the distributed cache includes nonvolatile memory.

[0064] In Example 13 the subject matter of any of Examples 9 or 10 optionally includes further comprising: periodically transferring information indicative of the data stored in the first local storage device to the distributed cache.

[0065] In Example 14 the subject matter of Example 13 optionally includes further comprising: transferring the information indicative of the data stored in the first local storage

device from the distributed cache to a third local storage device in direct communication with a third node in response to a determination that the first node has experienced a failure.

[0066] Example 15 is a machine readable medium, on which are stored instructions, comprising instructions that when executed cause a machine to: determine whether a size of data stored in a first local storage device in communication with a first node is greater than or equal to a predetermined threshold; and transfer a portion of the data stored in the first local storage device to a second local storage device in direct communication with a second node via a distributed cache in response to a determination that the size of the data stored in the first local storage device is greater than or equal to the predetermined threshold.

[0067] In Example 16 the subject matter of Example 15 optionally includes wherein the predetermined threshold is a ratio of the size of the data stored in a first local storage device relative to a capacity of the first local storage device.

[0068] In Example 17 the subject matter of any of Examples 15 or 16 optionally includes wherein the first local storage device includes volatile memory.

[0069] In Example 18 the subject matter of any of Examples 15 or 16 optionally includes wherein the distributed cache includes nonvolatile memory.

[0070] In Example 19 the subject matter of any of Examples 15 or 16 optionally includes further comprising instructions that when executed cause the machine to: periodically transfer information indicative of the data stored in the first local storage device to the distributed cache.

[0071] In Example 20 the subject matter of any of Examples 15 or 16 optionally includes further comprising instructions that when executed cause the machine to: transfer the information indicative of the data stored in the first local storage device from the distributed cache to a third local storage device in direct communication with a third node in response to a determination that the first node has experienced a failure.

[0072] Example 21 is an apparatus comprising: a processor; and a memory coupled with the processor, on which are stored instructions, comprising instruction that when executed cause the processor to: determine whether a size of data stored in a first local storage device is greater than or equal to a predetermined threshold; and transfer a portion of the data stored in the first local storage device to a second local storage device in direct communication with a second node via a distributed cache in response to a determination that the size of the data stored in the first local storage device is greater than or equal to the predetermined threshold.

[0073] In Example 22 the subject matter of Example 21 optionally includes wherein the predetermined threshold is a ratio of the size of the data stored in a first local storage device relative to a capacity of the first local storage device.

[0074] In Example 23 the subject matter of any of Examples 21 or 22 optionally includes wherein the first local storage device includes volatile memory.

[0075] In Example 24 the subject matter of any of Examples 21 or 22 optionally includes wherein the distributed cache includes nonvolatile memory.

[0076] In Example 25 the subject matter of any of Examples 21 or 22 optionally includes further comprising instructions that when executed cause the processor to: periodically transfer information indicative of the data stored in the first local storage device to the distributed cache.

[0077] In Example 26 the subject matter of any of Examples 21 or 22 optionally includes further comprising instructions that when executed cause the processor to: transfer the information indicative of the data stored in the first local storage device from the distributed cache to a third local storage device in direct communication with a third node in response to a determination that the first local storage device has experienced a failure.

[0078] It is to be understood that the above description is intended to be illustrative, and not restrictive. For example, the above-described embodiments may be used in combination with each other. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description. The scope of the invention therefore should be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

CLAIMS

What is claimed is:

1. A system, comprising:

a first node that processes data stored in a first local storage device;

a second node including a second local storage device; and

a distributed cache in network communication with the first node and the second node, wherein the system is configured to:

determine whether a size of the data stored in the first local storage device is greater than or equal to a predetermined threshold; and

transfer a portion of the data stored in the first local storage device to the second local storage device via the distributed cache in response to a determination that the size of the data stored in the first local storage device is greater than or equal to the predetermined threshold.

2. The system of claim 1, wherein the predetermined threshold is a ratio of the size of the data stored in the first local storage device relative to a capacity of the first local storage device.

3. The system of claims 1 or 2, wherein the first local storage device is in direct communication with the first node.

4. The system of claims 1 or 2, wherein the first local storage device includes volatile memory.

5. The system of claims 1 or 2, wherein the distributed cache includes nonvolatile memory.

6. The system of claims 1 or 2, further comprising a controller that outputs instructions to the first node, the second node, and the distributed cache.

7. The system of claims 1 or 2, wherein the first node periodically outputs information indicative of the data stored in the first local storage device to the distributed cache.

8. The system of claim 7, wherein the system is further configured to:

determine whether the first node has experienced a failure; and

transfer the information indicative of the data stored in the first local storage device from the distributed cache to the second local storage device in response to a determination that the first node has experienced a failure.

9. A method, comprising:

storing data in a first local storage device in communication with a first node;

determining whether a size of the data stored in the first local storage device is greater than or equal to a predetermined threshold; and

transferring a portion of the data stored in the first local storage device to a second local storage device in direct communication with a second node via a distributed cache in response to a determination that the size of the data stored in the first local storage device is greater than or equal to the predetermined threshold.

10. The method of claim 9, wherein the predetermined threshold is a ratio of the size of the data stored in the first local storage device relative to a capacity of the first local storage device.

11. The method of claims 9 or 10, wherein the first local storage device includes volatile memory.

12. The method of claims 9 or 10, wherein the distributed cache includes nonvolatile memory.

13. The method of claims 9 or 10, further comprising:

periodically transferring information indicative of the data stored in the first local storage device to the distributed cache.

14. The method of claim 13, further comprising:

transferring the information indicative of the data stored in the first local storage device from the distributed cache to a third local storage device in direct communication

with a third node in response to a determination that the first node has experienced a failure.

15. A machine readable medium, on which are stored instructions, comprising instructions that when executed cause a machine to:

determine whether a size of data stored in a first local storage device in communication with a first node is greater than or equal to a predetermined threshold; and

transfer a portion of the data stored in the first local storage device to a second local storage device in direct communication with a second node via a distributed cache in response to a determination that the size of the data stored in the first local storage device is greater than or equal to the predetermined threshold.

16. The machine readable medium of claim 15, wherein the predetermined threshold is a ratio of the size of the data stored in a first local storage device relative to a capacity of the first local storage device.

17. The machine readable medium of claims 15 or 16, wherein the first local storage device includes volatile memory.

18. The machine readable medium of claims 15 or 16, wherein the distributed cache includes nonvolatile memory.

19. The machine readable medium of claims 15 or 16, further comprising instructions that when executed cause the machine to:

periodically transfer information indicative of the data stored in the first local storage device to the distributed cache.

20. The machine readable medium of claims 15 or 16, further comprising instructions that when executed cause the machine to:

transfer the information indicative of the data stored in the first local storage device from the distributed cache to a third local storage device in direct communication

with a third node in response to a determination that the first node has experienced a failure.

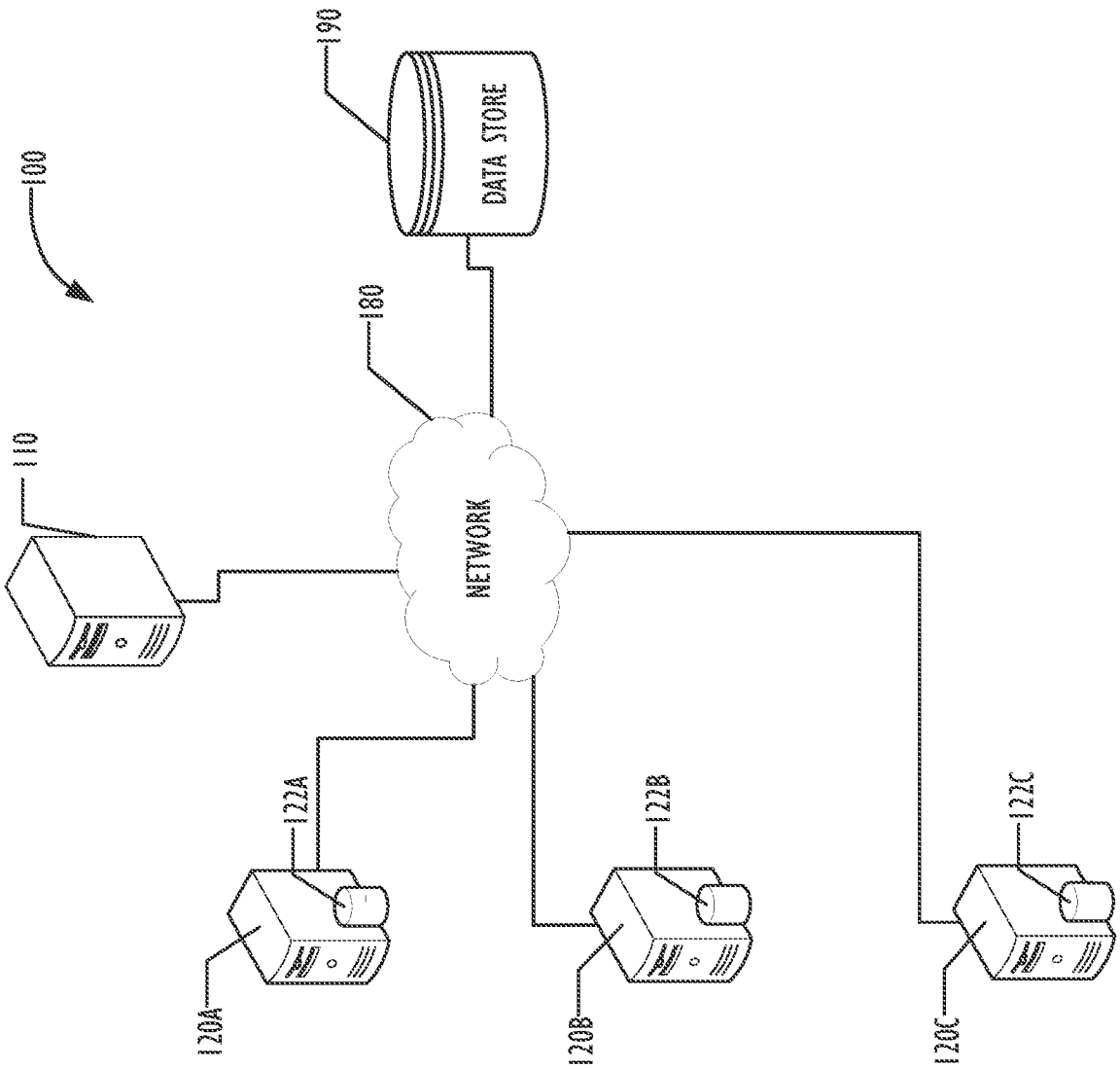
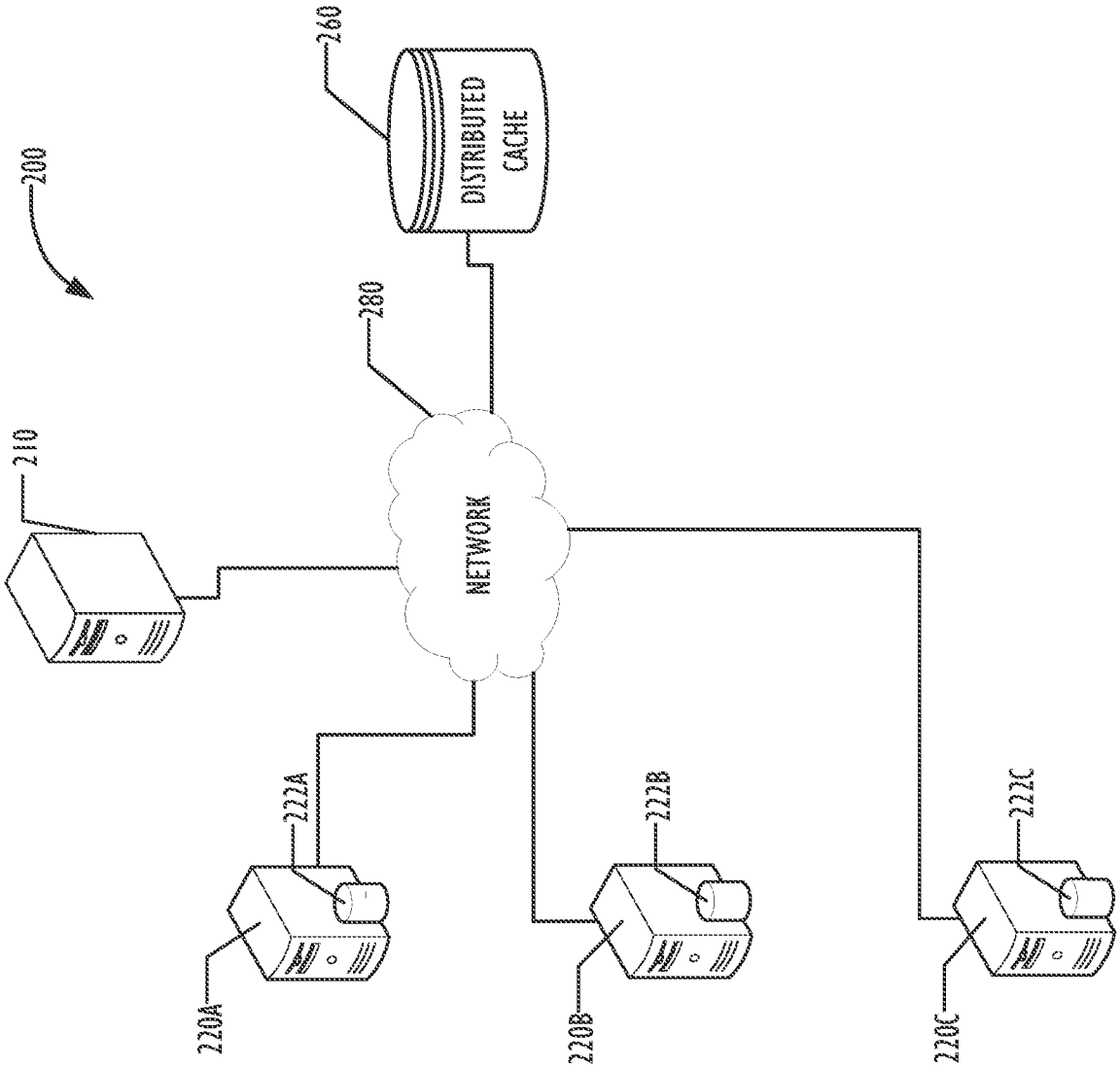


FIG. 1
(PRIOR ART)

FIG. 2


200
260
DISTRIBUTED CACHE
280
NETWORK
210
222A
222B
222C
220A
220B
220C

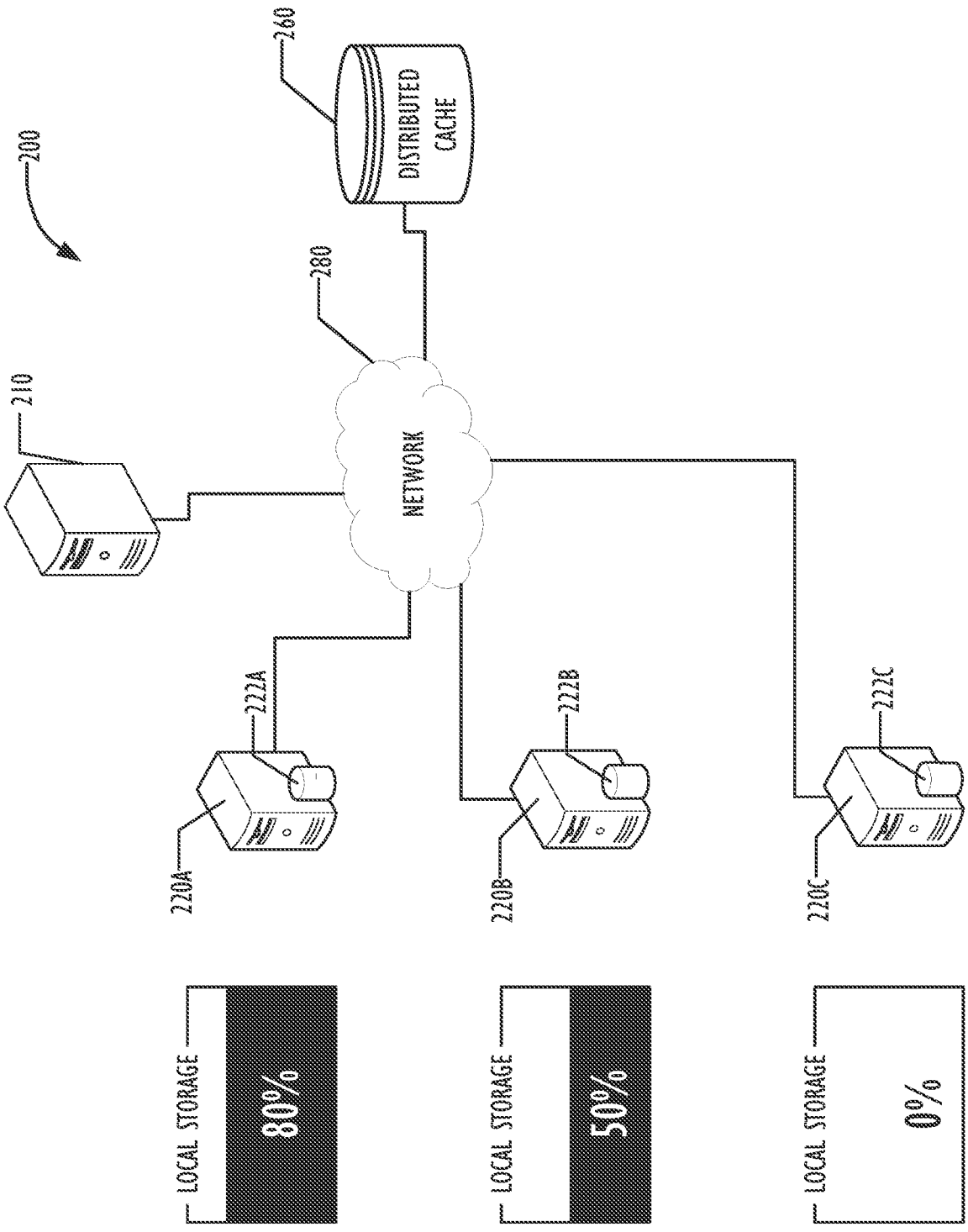

200
260
DISTRIBUTED CACHE
280
NETWORK
210
222A
222B
222C
220A
220B
220C
LOCAL STORAGE
80%
LOCAL STORAGE
50%
LOCAL STORAGE
0%

FIG. 3

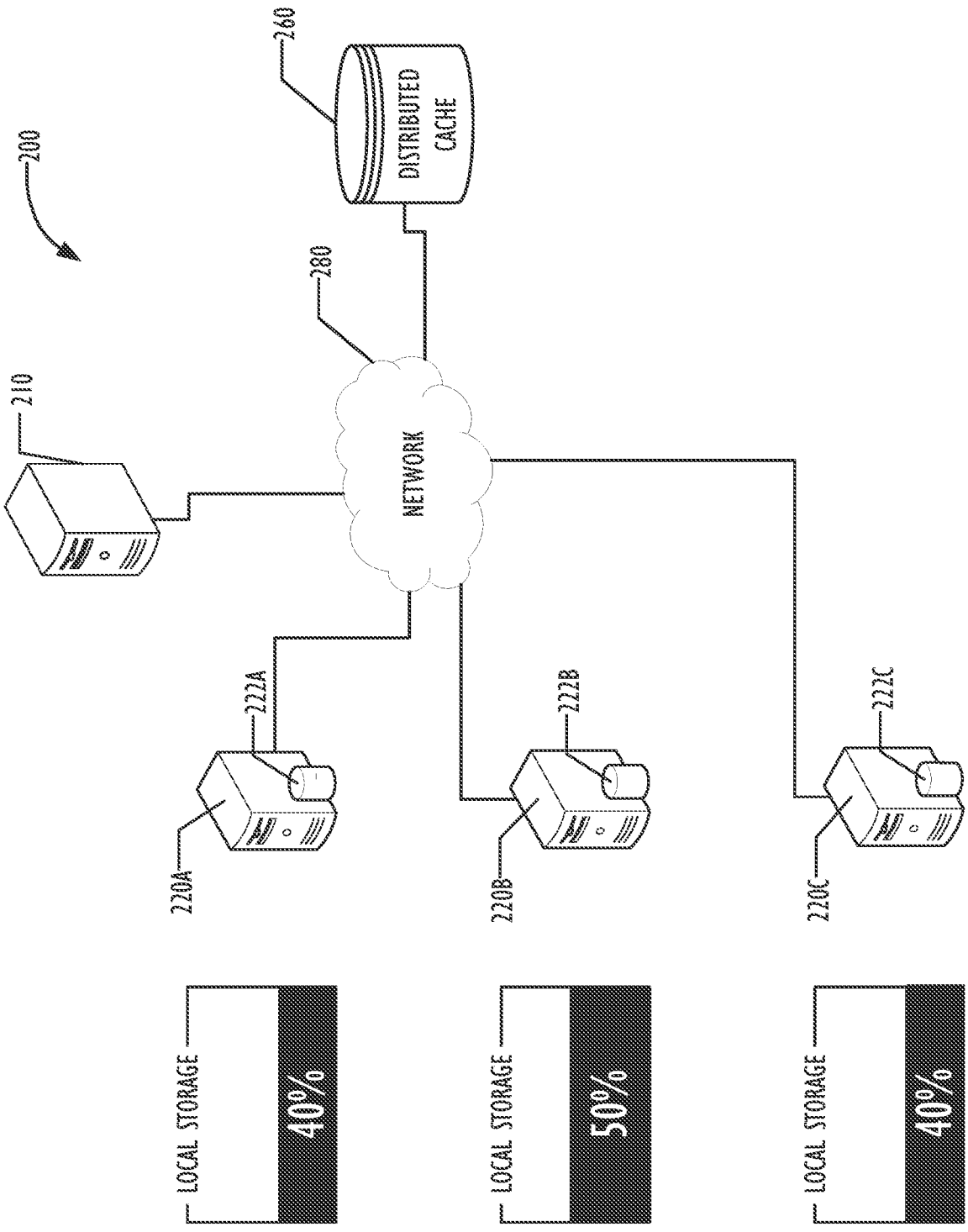

200
260
DISTRIBUTED CACHE
280
NETWORK
210
222A
222B
222C
220A
220B
220C
LOCAL STORAGE
40%
LOCAL STORAGE
50%
LOCAL STORAGE
40%

FIG. 4

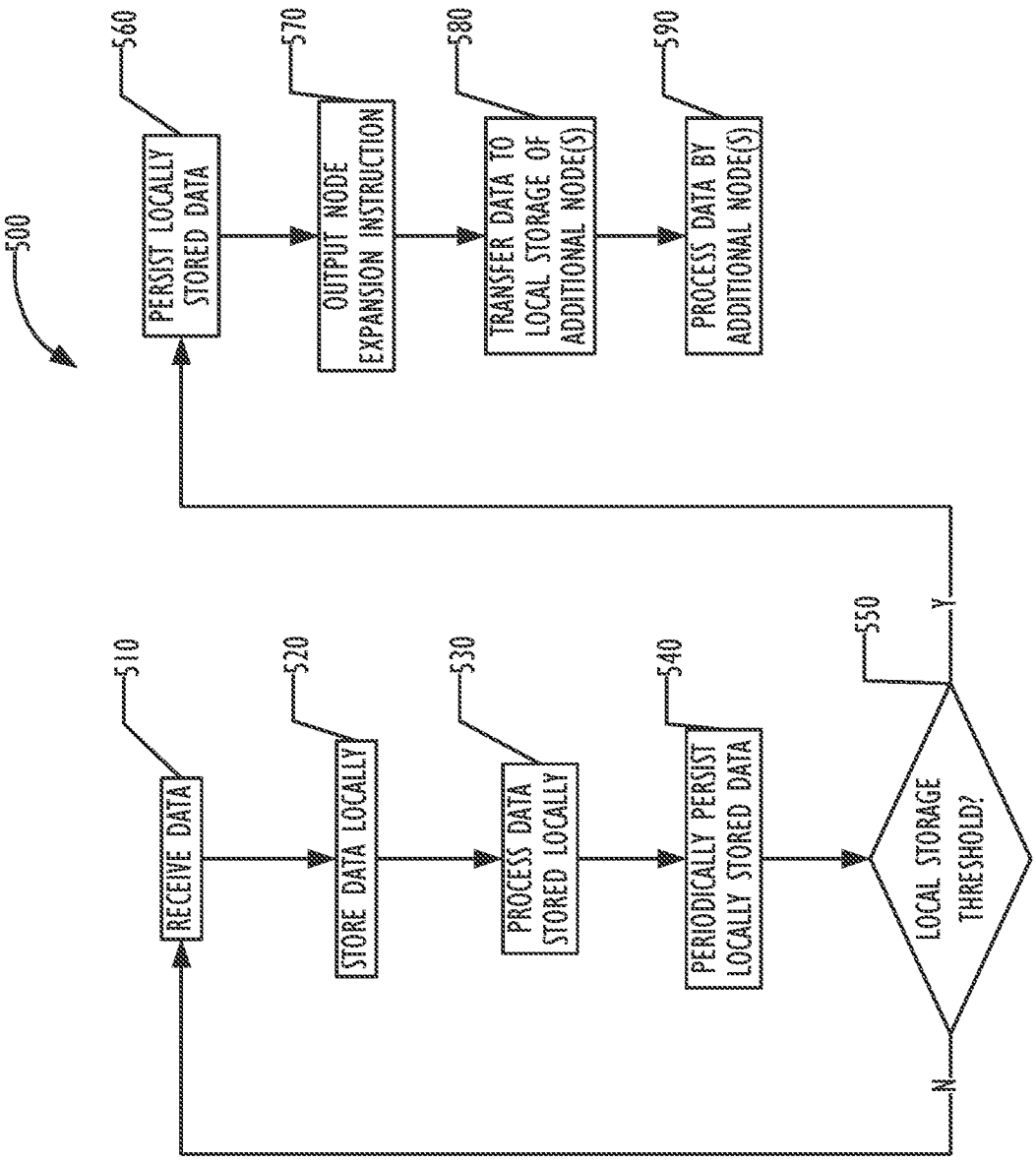

500
510
RECEIVE DATA
520
STORE DATA LOCALLY
530
PROCESS DATA STORED LOCALLY
540
PERIODICALLY PERSIST LOCALLY STORED DATA
550
LOCAL STORAGE THRESHOLD?
Y
N
560
PERSIST LOCALLY STORED DATA
570
OUTPUT NODE EXPANSION INSTRUCTION
580
TRANSFER DATA TO LOCAL STORAGE OF ADDITIONAL NODE(S)
590
PROCESS DATA BY ADDITIONAL NODE(S)

FIG. 5

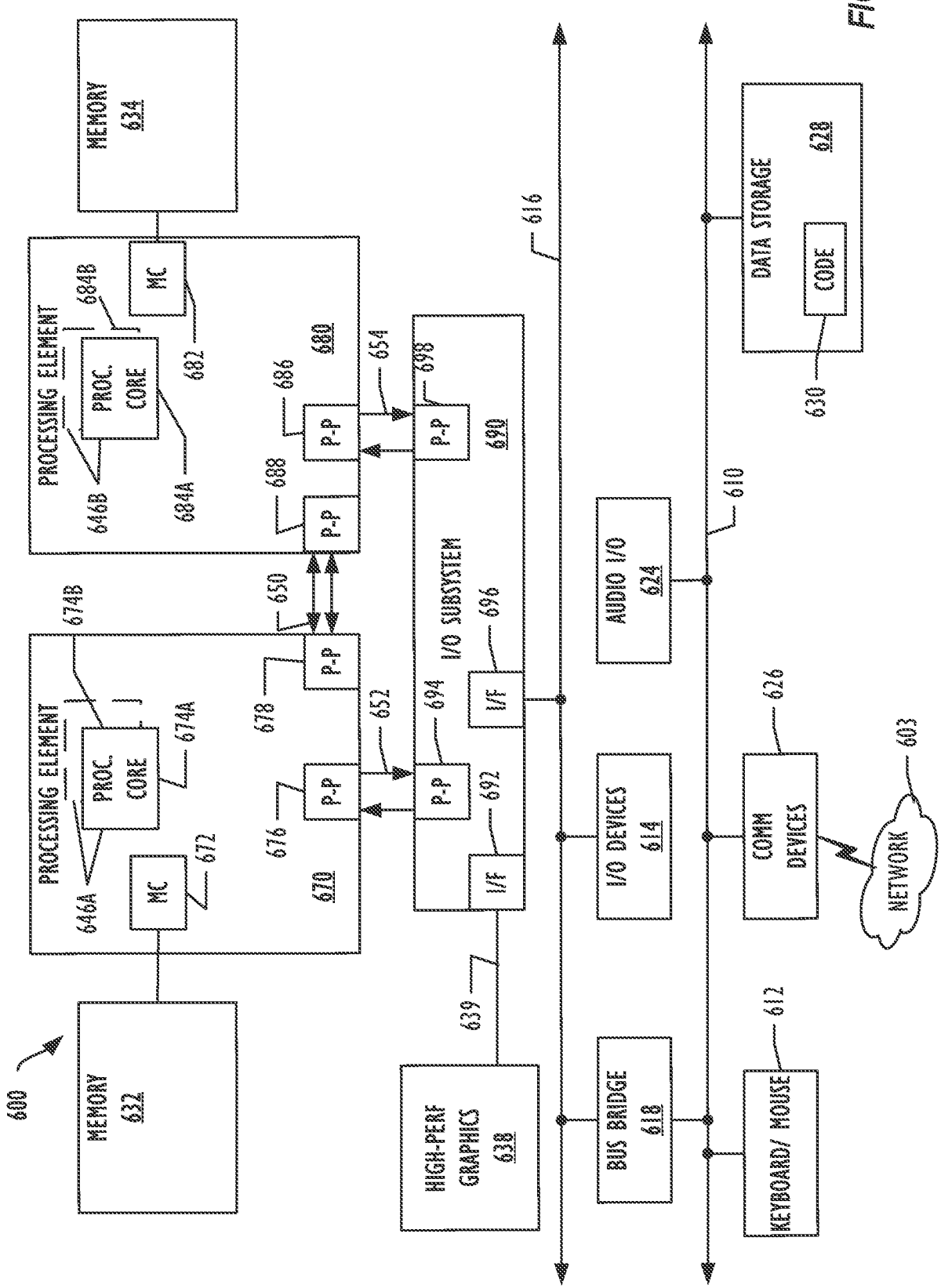

600
MEMORY 632
MEMORY 634
PROCESSING ELEMENT
PROC. CORE
646A
646B
MC
672
674A
674B
682
684A
684B
686
688
680
654
650
652
670
676
678
694
692
696
690
698
I/F
I/O SUBSYSTEM
616
610
AUDIO I/O 624
I/O DEVICES 614
BUS BRIDGE 618
HIGH-PERF GRAPHICS 638
639
DATA STORAGE 628
CODE
630
626
COMM DEVICES
NETWORK
603
612
KEYBOARD/ MOUSE

FIG. 6

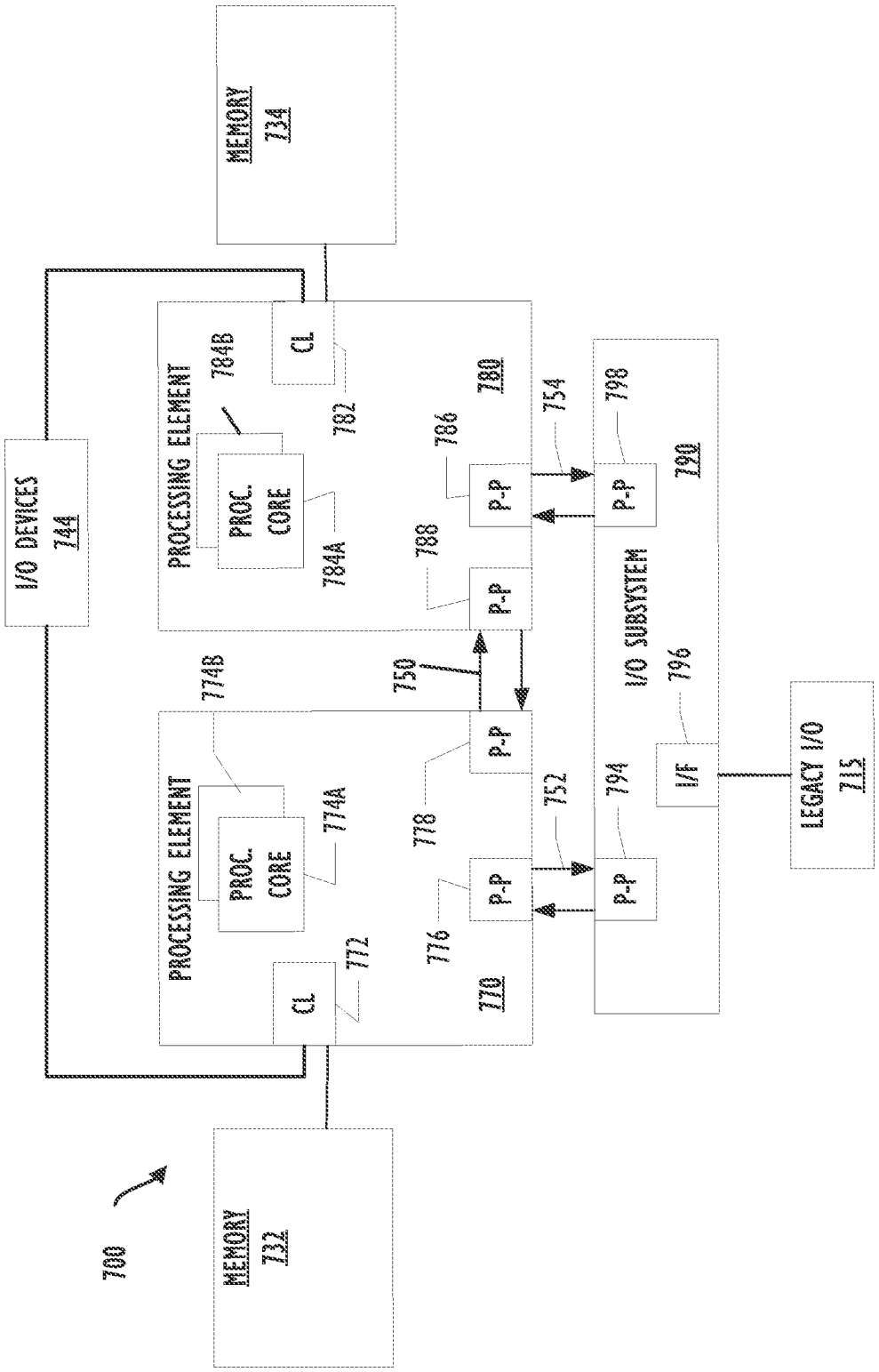

700
MEMORY 732
MEMORY 734
I/O DEVICES 744
PROCESSING ELEMENT
PROC. CORE
CL
772
774A
774B
770
776
778
P-P
750
752
754
780
782
784A
784B
786
788
790
794
796
798
I/O SUBSYSTEM
I/F
LEGACY I/O 713

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/055144

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/0813(2016.01)i, G06F 17/30(2006.01)i

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 12/0813; G06F 3/06; G06F 13/10; G06F 11/10; G06F 12/00; G06F 1/32; G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: local storage, usage, distributed system, transfer data, optimizing

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004-0039891 A1 (ALBERT LEUNG et al.) 26 February 2004 See paragraphs [0031]-[0033], [0044]-[0047], [0057], [0065], [0103]-[0123]; claim 1; and figures 1, 4.	1-20
A	US 2012-0137087 A1 (KIYOKAZU UMIMURA) 31 May 2012 See paragraphs [0077]-[0081], [0099]-[0102]; claim 3; and figure 5.	1-20
A	US 9026760 B1 (EMC CORPORATION) 05 May 2015 See column 16, line 54 - column 18, line 21; and figures 6-9.	1-20
A	US 2009-0049320 A1 (WILLIAM P. DAWKINS et al.) 19 February 2009 See claim 1; and figure 2.	1-20
A	US 2015-0149719 A1 (GARY STEPHEN SHUSTER) 28 May 2015 See claims 2, 8-10; and figures 3-4.	1-20

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

☒ 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
24 January 2017 (24.01.2017)	**25 January 2017 (25.01.2017)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/055144

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004-0039891 A1	26/02/2004	US 2003-0046270 A1	06/03/2003
		US 2004-0054656 A1	18/03/2004
		US 2005-0033757 A1	10/02/2005
		US 2007-0083575 A1	12/04/2007
		US 7092977 B2	15/08/2006
		US 7454446 B2	18/11/2008
		US 7509316 B2	24/03/2009
US 2012-0137087 A1	31/05/2012	JP 2012-118587 A	21/06/2012
		JP 5641900 B2	17/12/2014
US 9026760 B1	05/05/2015	None	
US 2009-0049320 A1	19/02/2009	CN 101369219 A	18/02/2009
		CN 101369219 B	06/04/2011
		EP 2026185 A2	18/02/2009
		EP 2026185 A3	25/03/2009
		EP 2026185 B1	08/08/2012
		SG 150445 A1	30/03/2009
		TW 200925977 A	16/06/2009
		TW I377500 B	21/11/2012
		US 8046597 B2	25/10/2011
US 2015-0149719 A1	28/05/2015	US 2008-0275928 A1	06/11/2008
		US 2011-0238912 A1	29/09/2011
		US 2016-0019115 A1	21/01/2016
		US 7958303 B2	07/06/2011
		US 8819365 B2	26/08/2014
		US 9405627 B2	02/08/2016
		US 9448886 B2	20/09/2016