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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 July 2016 (28.07.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/118564 A1

(51) International Patent Classification:
G06F 13/00 (2006.01)

(21) International Application Number:
PCT/US2016/014024

(22) International Filing Date:
20 January 2016 (20.01.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/105,482 20 January 2015 (20.01.2015) US
15/001,340 20 January 2016 (20.01.2016) US

(71) Applicant: **ULTRATA LLC** [US/US]; 1934 Old Gallows Road, Suite 350, Vienna, Virginia 22182 (US).

(72) Inventors: **FRANK, Steven**; 1804 Walnut Hollow Lane, Boulder, Colorado 80302 (US). **REBACK, Larry**; 1807 Brooktrail Court, Vienna, Virginia 22182 (US).

(74) Agents: **DALEY, William J.** et al.; Two Embarcadero Center, Eighth Floor, San Francisco, California 94111 (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).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

Published:

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

(54) Title: UNIVERSAL SINGLE LEVEL OBJECT MEMORY ADDRESS SPACE

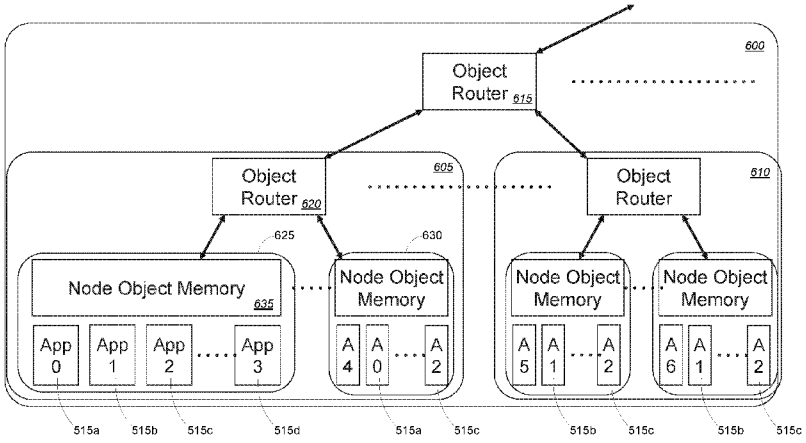

FIG. 6

(57) Abstract: Embodiments of the invention provide systems and methods for managing processing, memory, storage, network, and cloud computing to significantly improve the efficiency and performance of processing nodes. Embodiments described herein can eliminate typical size constraints on memory space of commodity servers and other commodity hardware imposed by address sizes. Rather, physical addressing can be managed within the memory objects themselves and the objects can be in turn accessed and managed through the object name space.

WO 2016/118564 A1

UNIVERSAL SINGLE LEVEL OBJECT MEMORY ADDRESS SPACE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application claims benefit under 35 USC 119(e) of U.S. Provisional Application No. 62/105,482, filed on January 20, 2015 by Frank et al and entitled "Infinite Memory Fabric Architecture," of which the entire disclosure is incorporated herein by reference for all purposes.

[0002] The present application is also related to the following co-pending and commonly assigned U.S. Patent Applications:

[0003] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967319(000100US)) filed concurrent herewith by Frank and entitled "Object Based Memory Fabric;"

[0004] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967320(000110US)) filed concurrent herewith by Frank and entitled "Trans-Cloud Object Based Memory;"

[0005] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967322(000130US)) filed concurrent herewith by Frank and entitled "Object Memory Fabric Performance Acceleration;"

[0006] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967323(000200US)) filed concurrent herewith by Frank and entitled "Distributed Index for Fault Tolerance Object Memory Fabric;"

[0007] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967324(000210US)) filed concurrent herewith by Frank and entitled "Implementation of an Object Memory Centric Cloud;"

[0008] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967325(000220US)) filed concurrent herewith by Frank and entitled "Managing Metadata in an Object Memory Fabric;"

[0009] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967326(000230US)) filed concurrent herewith by Frank and entitled "Utilization of a Distributed Index to Provide Object Memory Fabric Coherency;"

[0010] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967327(000300US)) filed concurrent herewith by Frank and entitled "Object Memory Data Flow Instruction Execution;"

[0011] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967329(000310US)) filed concurrent herewith by Frank and entitled "Object Memory Data Flow Triggers;" and

[0012] U.S. Patent Application No. _____ (Attorney Docket Number 097704-0967328(000320US)) filed concurrent herewith by Frank and entitled "Object Memory Instruction Set," of which the entire disclosure of each is incorporated herein by reference for all purposes.

BACKGROUND OF THE INVENTION

[0013] Embodiments of the present invention relate generally to methods and systems for improving performance of processing nodes in a fabric and more particularly to changing the way in which processing, memory, storage, network, and cloud computing, are managed to significantly improve the efficiency and performance of commodity hardware.

[0014] As the size and complexity of data and the processes performed thereon continually increases, computer hardware is challenged to meet these demands. Current commodity hardware and software solutions from established server, network and storage providers are unable to meet the demands of Cloud Computing and Big Data environments. This is due, at

least in part, to the way in which processing, memory, and storage are managed by those systems. Specifically, processing is separated from memory which is turn is separated from storage in current systems and each of processing, memory, and storage is managed separately by software. Each server and other computing device (referred to herein as a node) is in turn separated from other nodes by a physical computer network, managed separately by software and in turn the separate processing, memory, and storage associated with each node are managed by software on that node.

[0015] FIG. 1 is a block diagram illustrating an example of the separation data storage, memory, and processing within prior art commodity servers and network components. This example illustrates a system 100 in which commodity servers 105 and 110 are communicatively coupled with each other via a physical network 115 and network software 155 as known in the art. Also as known in the art, the servers can each execute any number of one or more applications 120a, 120b, 120c of any variety. As known in the art, each application 120a, 120b, 120c executes on a processor (not shown) and memory (not shown) of the server 105 and 110 using data stored in physical storage 150. Each server 105 and 110 maintains a directory 125 mapping the location of the data used by the applications 120a, 120b, 120c. Additionally, each server implements for each executing application 120a, 120b, 120c a software stack which includes an application representation 130 of the data, a database representation 135, a file system representation 140, and a storage representation 145.

[0016] While effective, there are three reasons that such implementations on current commodity hardware and software solutions from established server, network and storage providers are unable to meet the increasing demands of Cloud Computing and Big Data environments. One reason for the shortcomings of these implementations is their complexity. The software stack must be in place and every application must manage the separation of storage, memory, and processing as well as applying parallel server resources. Each application must trade-off algorithm parallelism, data organization and data movement which is extremely challenging to get correct, let alone considerations of performance and economics. This tends to lead to implementation of more batch oriented solutions in the applications, rather than the integrated real-time solutions preferred by most businesses. Additionally, separation of storage,

memory, and processing, in such implementations also creates significant inefficiency for each layer of the software stack to find, move, and access a block of data due to the required instruction execution and latencies of each layer of the software stack and between the layers. Furthermore, this inefficiency limits the economic scaling possible and limits the data-size for all but the most extremely parallel algorithms. The reason for the latter is that the efficiency with which servers (processors or threads) can interact limits the amount of parallelism due to Amdahl's law. Hence, there is a need for improved methods and systems for managing processing, memory, and storage to significantly improve the performance of processing nodes.

BRIEF SUMMARY OF THE INVENTION

[0017] Embodiments of the invention provide systems and methods for managing processing, memory, storage, network, and cloud computing to significantly improve the efficiency and performance of processing nodes. Embodiments described herein can eliminate typical size constraints on memory space of commodity servers and other commodity hardware imposed by address sizes. Rather, physical addressing can be managed within the memory objects themselves and the objects can be in turn accessed and managed through the object name space.

[0018] According to one embodiment, a hardware-based processing node of an object memory fabric can comprise a memory module storing and managing one or more memory objects. Each memory object can be created natively within the memory module, can be accessed using a single memory reference instruction without Input/Output (I/O) instructions, and can be managed by the memory module at a single memory layer. Physical addressing of both memory and storage of the object memory fabric can be managed with each of the one or more memory objects through an object name space of the object memory fabric. The object name space is unconstrained by the physical addresses managed by the one or more memory objects. The memory module can manage both storage and memory without distinction through the one or more memory objects. Managing both storage and memory without distinction through the one or more memory objects can comprise managing all of the one or more memory objects as memory regardless of an underlying physical storage media. The object memory fabric can comprise a plurality of hardware-based processing nodes and the one or more memory objects can be accessed and managed across the object memory fabric through the object name space of

the object memory fabric. Each memory object and properties of each memory object can be maintained on any one or more of the plurality of nodes in the object memory fabric and managing the memory objects can include maintaining the memory objects and properties of the memory objects as the memory objects are moved, split, or duplicated between nodes

[0019] In some cases, the hardware-based processing node can comprise a Dual In-line Memory Module (DIMM) card. For example, the hardware-based processing node can comprise a commodity server and the memory module can comprise a Dual In-line Memory Module (DIMM) card installed within the commodity server. A communication interface can also be coupled with the object memory fabric. For example, the communication interface comprises a Peripheral Component Interconnect Express (PCI-e) card. In other cases, the hardware-based processing node can comprise a mobile computing device. In yet another example, the hardware-based processing node can comprise a single chip.

[0020] According to one embodiment, an object memory fabric can comprise a plurality of hardware-based processing nodes. Each hardware-based processing node can comprise one or more memory modules storing and managing one or more memory objects, wherein each memory object is created natively within the memory module, each memory object is accessed using a single memory reference instruction without Input/Output (I/O) instructions, and each memory object is managed by the memory module at a single memory layer. Physical addressing of both memory and storage of the object memory fabric can be managed with each of the one or more memory objects through an object name space of the object memory fabric. The memory module can manage both storage and memory without distinction through the one or more memory objects. Managing both storage and memory without distinction through the one or more memory objects can comprise managing all of the one or more memory objects as memory regardless of an underlying physical storage media. Each hardware-based processing node can also comprise a node router communicatively coupled with each of the one or more memory modules of the node and adapted to route memory objects or portions of memory objects between the one or more memory modules of the node. The object memory fabric can further comprise one or more inter-node routers communicatively coupled with each node router, wherein each of the plurality of nodes of the object memory fabric is communicatively coupled

with at least one of the inter-node routers and adapted to route memory objects or portions of memory objects between the plurality of nodes.

[0021] The one or more memory objects can be accessed and managed across the object memory fabric through the object name space of the object memory fabric. The object name space is unconstrained by the physical addresses managed by the one or more memory objects. Each memory object and properties of each memory object can be maintained on any one or more of the plurality of nodes in the object memory fabric and managing the memory objects can include maintaining the memory objects and properties of the memory objects as the memory objects are moved, split, or duplicated between nodes. In some implementations, at least one hardware-based processing node can comprise a commodity server, the one or more memory modules of the commodity server can comprise at least one Dual In-line Memory Module (DIMM) card installed within the commodity server. In such cases, the communication interface can comprise a Peripheral Component Interconnect Express (PCI-e) card. Additionally or alternatively, at least one hardware-based processing node can comprise a mobile computing device, a single chip, and/or other form factor.

[0022] According to yet another embodiment, a method for storing and managing one or more memory objects in an object memory fabric can comprise creating each memory object natively within a memory module of a hardware-based processing node of the object memory fabric, accessing each memory object using a single memory reference instruction without Input/Output (I/O) instructions, managing each memory object within the memory module at a single memory layer, and managing physical addressing of both memory and storage of the object memory fabric with each of the one or more memory objects through an object name space of the object memory fabric. Both storage and memory can be managed by the memory module without distinction through the one or more memory objects and the object name space is unconstrained by the physical addresses managed by the one or more memory objects. Managing both storage and memory without distinction through the one or more memory objects can comprise managing all of the one or more memory objects as memory regardless of an underlying physical storage media. The object memory fabric can comprise a plurality of hardware-based processing nodes and the one or more memory objects can be accessed and managed across the object

memory fabric through the object name space of the object memory fabric. Each memory object and properties of each memory object can be maintained on any one or more of the plurality of nodes in the object memory fabric and managing the memory objects can include maintaining the memory objects and properties of the memory objects as the memory objects are moved, split, or duplicated between nodes.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] FIG. 1 is a block diagram illustrating an example of the separation data storage, memory, processing, network, and cloud computing within prior art commodity servers and network components.

[0024] FIG. 2 is a block diagram illustrating components of an exemplary distributed system in which various embodiments of the present invention may be implemented.

[0025] FIG. 3 is a block diagram illustrating an exemplary computer system in which embodiments of the present invention may be implemented.

[0026] FIG. 4 is a block diagram illustrating an exemplary object memory fabric architecture according to one embodiment of the present invention.

[0027] FIG. 5 is a block diagram illustrating an exemplary memory fabric object memory according to one embodiment of the present invention.

[0028] FIG. 6 is a block diagram illustrating an exemplary object memory dynamics and physical organization according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0029] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of various embodiments of the present invention. It will be apparent, however, to one skilled in the art that embodiments of the present

invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form.

[0030] The ensuing description provides exemplary embodiments only, and is not intended to limit the scope, applicability, or configuration of the disclosure. Rather, the ensuing description of the exemplary embodiments will provide those skilled in the art with an enabling description for implementing an exemplary embodiment. It should be understood that various changes may be made in the function and arrangement of elements without departing from the spirit and scope of the invention as set forth in the appended claims.

[0031] Specific details are given in the following description to provide a thorough understanding of the embodiments. However, it will be understood by one of ordinary skill in the art that the embodiments may be practiced without these specific details. For example, circuits, systems, networks, processes, and other components may be shown as components in block diagram form in order not to obscure the embodiments in unnecessary detail. In other instances, well-known circuits, processes, algorithms, structures, and techniques may be shown without unnecessary detail in order to avoid obscuring the embodiments.

[0032] Also, it is noted that individual embodiments may be described as a process which is depicted as a flowchart, a flow diagram, a data flow diagram, a structure diagram, or a block diagram. Although a flowchart may describe the operations as a sequential process, many of the operations can be performed in parallel or concurrently. In addition, the order of the operations may be re-arranged. A process is terminated when its operations are completed, but could have additional steps not included in a figure. A process may correspond to a method, a function, a procedure, a subroutine, a subprogram, etc. When a process corresponds to a function, its termination can correspond to a return of the function to the calling function or the main function.

[0033] The term "machine-readable medium" includes, but is not limited to portable or fixed storage devices, optical storage devices, wireless channels and various other mediums capable of storing, containing or carrying instruction(s) and/or data. 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. Various other terms used herein are now defined for the sake of clarity.

[0034] Virtual memory is a memory management technique that gives the illusion to each software process that memory is as large as the virtual address space. The operating system in conjunction with differing degrees of hardware manages the physical memory as a cache of the virtual address space, which is placed in secondary storage and accessible through Input/Output instructions. Virtual memory is separate from, but can interact with, a file system.

[0035] A single level store is an extension of virtual memory in which there are no files, only persistent objects or segments which are mapped into a processes' address space using virtual memory techniques. The entire storage of the computing system is thought of as a segment and address within a segment. Thus at least three separate address spaces, i.e., physical memory address/node, virtual address/process, and secondary storage address/disk, are managed by software.

[0036] Object storage refers to the way units of storage called objects are organized. Every object consists of a container that holds three things: actual data; expandable metadata; and a globally unique identifier referred to herein as the object address. The metadata of the object is used to define contextual information about the data and how it should be used and managed including relationship to other objects.

[0037] The object address space is managed by software over storage devices, nodes, and network to find an object without knowing its physical location. Object storage is separate from virtual memory and single level store, but can certainly inter-operate through software.

[0038] Block storage consists of evenly sized blocks of data with an address based on a physical location and without metadata.

[0039] A network address is a physical address of a node within an IP network that is associated with a physical location.

[0040] A node or processing node is a physical unit of computing delineated by a shared physical memory that be addressed by any processor within the node.

[0041] Object memory is an object store directly accessible as memory by processor memory reference instructions and without implicit or explicit software or Input/Output instructions required. Object capabilities are directly provided within the object memory to processing through memory reference instructions.

[0042] An object memory fabric connects object memory modules and nodes into a single object memory where any object is local to any object memory module by direct management, in hardware, of object data, meta-data and object address.

[0043] An object router routes objects or portions of objects in an object memory fabric based on an object address. This is distinct from a conventional router which forwards data packets to appropriate part of a network based on a network address.

[0044] Embodiments may be implemented by hardware, software, firmware, middleware, microcode, hardware description languages, or any combination thereof. When implemented in software, firmware, middleware or microcode, the program code or code segments to perform the necessary tasks may be stored in a machine readable medium. A processor(s) may perform the necessary tasks.

[0045] Embodiments of the invention provide systems and methods for managing processing, memory, storage, network, and cloud computing to significantly improve the efficiency and performance of processing nodes. Embodiments described herein can be implemented in a set of hardware components that, in essence, change the way in which processing, memory, and

storage, network, and cloud computing are managed by breaking down the artificial distinctions between processing, memory, storage and networking in today's commodity solutions to significantly improve the efficiency and performance of commodity hardware. For example, the hardware elements can include a standard format memory module, such as a (DIMM) and a set of one or more object routers. The memory module can be added to commodity or "off-the-shelf" hardware such a server node and acts as a big data accelerator within that node. Object routers can be used to interconnect two or more servers or other nodes adapted with the memory modules and help to manage processing, memory, and storage across these different servers. Nodes can be physically close or far apart. Together, these hardware components can be used with commodity servers or other types of computing nodes in any combination to implement the embodiments described herein.

[0046] According to one embodiment, such hardware components can implement an object-based memory which manages the objects within the memory and at the memory layer rather than in the application layer. That is, the objects and associated properties are implemented and managed natively in memory enabling the object memory system to provide increased functionality without any software and increasing performance by dynamically managing object characteristics including, but not limited to persistence, location and processing. Object properties can also propagate up to higher application levels.

[0047] Such hardware components can also eliminate the distinction between memory (temporary) and storage (persistent) by implementing and managing both within the objects. These components can eliminate the distinction between local and remote memory by transparently managing the location of objects (or portions of objects) so all objects appear simultaneously local to all nodes. These components can also eliminate the distinction between processing and memory through methods of the objects to place the processing within the memory itself.

[0048] According to one embodiment, such hardware components can eliminate typical size constraints on memory space of the commodity servers imposed by address sizes. Rather, physical addressing can be managed within the memory objects themselves and the objects can in turn be accessed and managed through the object name space.

[0049] Embodiment described herein can provide transparent and dynamic performance acceleration, especially with big data or other memory intensive applications by reducing or eliminating overhead typically associated with memory management, storage management, networking and data directories. Rather, management of the memory objects at the memory level can significantly shorten the pathways between storage and memory and between memory and processing, thereby eliminating the associated overhead between each. Various additional details of embodiments of the present invention will be described below with reference to the figures.

[0050] FIG. 2 is a block diagram illustrating components of an exemplary distributed system in which various embodiments of the present invention may be implemented. In the illustrated embodiment, distributed system 200 includes one or more client computing devices 202, 204, 206, and 208, which are configured to execute and operate a client application such as a web browser, proprietary client, or the like over one or more network(s) 210. Server 212 may be communicatively coupled with remote client computing devices 202, 204, 206, and 208 via network 210.

[0051] In various embodiments, server 212 may be adapted to run one or more services or software applications provided by one or more of the components of the system. In some embodiments, these services may be offered as web-based or cloud services or under a Software as a Service (SaaS) model to the users of client computing devices 202, 204, 206, and/or 208. Users operating client computing devices 202, 204, 206, and/or 208 may in turn utilize one or more client applications to interact with server 212 to utilize the services provided by these components. For the sake of clarity, it should be noted that server 212 and database 214, 216 can correspond to server 105 described above with reference to FIG. 1. Network 210 can be part of or an extension to physical network 115. It should also be understood that there can be any number of client computing devices 202, 204, 206, 208 and servers 212, each with one or more databases 214, 216.

[0052] In the configuration depicted in the figure, the software components 218, 220 and 222 of system 200 are shown as being implemented on server 212. In other embodiments, one or more of the components of system 200 and/or the services provided by these components may

also be implemented by one or more of the client computing devices 202, 204, 206, and/or 208. Users operating the client computing devices may then utilize one or more client applications to use the services provided by these components. These components may be implemented in hardware, firmware, software, or combinations thereof. It should be appreciated that various different system configurations are possible, which may be different from distributed system 200. The embodiment shown in the figure is thus one example of a distributed system for implementing an embodiment system and is not intended to be limiting.

[0053] Client computing devices 202, 204, 206, and/or 208 may be portable handheld devices (e.g., an iPhone®, cellular telephone, an iPad®, computing tablet, a personal digital assistant (PDA)) or wearable devices (e.g., a Google Glass® head mounted display), running software such as Microsoft Windows Mobile®, and/or a variety of mobile operating systems such as iOS, Windows Phone, Android, BlackBerry 10, Palm OS, and the like, and being Internet, e-mail, short message service (SMS), Blackberry®, or other communication protocol enabled. The client computing devices can be general purpose personal computers including, by way of example, personal computers and/or laptop computers running various versions of Microsoft Windows®, Apple Macintosh®, and/or Linux operating systems. The client computing devices can be workstation computers running any of a variety of commercially-available UNIX® or UNIX-like operating systems, including without limitation the variety of GNU/Linux operating systems, such as for example, Google Chrome OS. Alternatively, or in addition, client computing devices 202, 204, 206, and 208 may be any other electronic device, such as a thin-client computer, an Internet-enabled gaming system (e.g., a Microsoft Xbox gaming console with or without a Kinect® gesture input device), and/or a personal messaging device, capable of communicating over network(s) 210.

[0054] Although exemplary distributed system 200 is shown with four client computing devices, any number of client computing devices may be supported. Other devices, such as devices with sensors, etc., may interact with server 212.

[0055] Network(s) 210 in distributed system 200 may be any type of network familiar to those skilled in the art that can support data communications using any of a variety of commercially-available protocols, including without limitation TCP/IP (Transmission Control Protocol/Internet

Protocol), SNA (Systems Network Architecture), IPX (Internet Packet Exchange), AppleTalk, and the like. Merely by way of example, network(s) 210 can be a Local Area Network (LAN), such as one based on Ethernet, Token-Ring and/or the like. Network(s) 210 can be a wide-area network and the Internet. It can include a virtual network, including without limitation a Virtual Private Network (VPN), an intranet, an extranet, a Public Switched Telephone Network (PSTN), an infra-red network, a wireless network (e.g., a network operating under any of the Institute of Electrical and Electronics (IEEE) 802.11 suite of protocols, Bluetooth®, and/or any other wireless protocol); and/or any combination of these and/or other networks. Elements of such networks can have an arbitrary distance, i.e., can be remote or co-located. Software Defined Networks (SDNs) can be implemented with a combination of dumb routers and software running on servers.

[0056] Server 212 may be composed of one or more general purpose computers, specialized server computers (including, by way of example, Personal Computer (PC) servers, UNIX® servers, mid-range servers, mainframe computers, rack-mounted servers, etc.), server farms, server clusters, or any other appropriate arrangement and/or combination. In various embodiments, server 212 may be adapted to run one or more services or software applications described in the foregoing disclosure. For example, server 212 may correspond to a server for performing processing described above according to an embodiment of the present disclosure.

[0057] Server 212 may run an operating system including any of those discussed above, as well as any commercially available server operating system. Server 212 may also run any of a variety of additional server applications and/or mid-tier applications, including HyperText Transport Protocol (HTTP) servers, File Transfer Protocol (FTP) servers, Common Gateway Interface (CGI) servers, JAVA® servers, database servers, and the like. Exemplary database servers include without limitation those commercially available from Oracle, Microsoft, Sybase, International Business Machines (IBM), and the like.

[0058] In some implementations, server 212 may include one or more applications to analyze and consolidate data feeds and/or event updates received from users of client computing devices 202, 204, 206, and 208. As an example, data feeds and/or event updates may include, but are not limited to, Twitter® feeds, Facebook® updates or real-time updates received from one or more

third party information sources and continuous data streams, which may include real-time events related to sensor data applications, financial tickers, network performance measuring tools (e.g., network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like. Server 212 may also include one or more applications to display the data feeds and/or real-time events via one or more display devices of client computing devices 202, 204, 206, and 208.

[0059] Distributed system 200 may also include one or more databases 214 and 216. Databases 214 and 216 may reside in a variety of locations. By way of example, one or more of databases 214 and 216 may reside on a non-transitory storage medium local to (and/or resident in) server 212. Alternatively, databases 214 and 216 may be remote from server 212 and in communication with server 212 via a network-based or dedicated connection. In one set of embodiments, databases 214 and 216 may reside in a Storage-Area Network (SAN). Similarly, any necessary files for performing the functions attributed to server 212 may be stored locally on server 212 and/or remotely, as appropriate. In one set of embodiments, databases 214 and 216 may include relational databases that are adapted to store, update, and retrieve data in response to commands, e.g., MySQL-formatted commands. Additionally or alternatively, server 212 can provide and support big data processing on unstructured data including but not limited to Hadoop processing, NoSQL databases, graph databases etc. In yet other implementations, server 212 may perform non-database types of bog data applications including but not limited to machine learning.

[0060] FIG. 3 is a block diagram illustrating an exemplary computer system in which embodiments of the present invention may be implemented. The system 300 may be used to implement any of the computer systems described above. As shown in the figure, computer system 300 includes a processing unit 304 that communicates with a number of peripheral subsystems via a bus subsystem 302. These peripheral subsystems may include a processing acceleration unit 306, an I/O subsystem 308, a storage subsystem 318 and a communications subsystem 324. Storage subsystem 318 includes tangible computer-readable storage media 322 and a system memory 310.

[0061] Bus subsystem 302 provides a mechanism for letting the various components and subsystems of computer system 300 communicate with each other as intended. Although bus subsystem 302 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple buses. Bus subsystem 302 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. For example, such architectures may include an Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, Peripheral Component Interconnect (PCI) bus, which can be implemented as a Mezzanine bus manufactured to the IEEE P1386.1 standard, or PCI enhanced (PCIe) bus.

[0062] Processing unit 304, which can be implemented as one or more integrated circuits (e.g., a conventional microprocessor or microcontroller), controls the operation of computer system 300. One or more processors may be included in processing unit 304. These processors may include single core or multicore processors. In certain embodiments, processing unit 304 may be implemented as one or more independent processing units 332 and/or 334 with single or multicore processors included in each processing unit. In other embodiments, processing unit 304 may also be implemented as a quad-core processing unit formed by integrating two dual-core processors into a single chip.

[0063] In various embodiments, processing unit 304 can execute a variety of programs in response to program code and can maintain multiple concurrently executing programs or processes. At any given time, some or all of the program code to be executed can be resident in processor(s) 304 and/or in storage subsystem 318. Through suitable programming, processor(s) 304 can provide various functionalities described above. Computer system 300 may additionally include a processing acceleration unit 306, which can include a Digital Signal Processor (DSP), a special-purpose processor, and/or the like.

[0064] I/O subsystem 308 may include user interface input devices and user interface output devices. User interface input devices may include a keyboard, pointing devices such as a mouse or trackball, a touchpad or touch screen incorporated into a display, a scroll wheel, a click wheel, a dial, a button, a switch, a keypad, audio input devices with voice command recognition

systems, microphones, and other types of input devices. User interface input devices may include, for example, motion sensing and/or gesture recognition devices such as the Microsoft Kinect® motion sensor that enables users to control and interact with an input device, such as the Microsoft Xbox® 360 game controller, through a natural user interface using gestures and spoken commands. User interface input devices may also include eye gesture recognition devices such as the Google Glass® blink detector that detects eye activity (e.g., 'blinking' while taking pictures and/or making a menu selection) from users and transforms the eye gestures as input into an input device (e.g., Google Glass®). Additionally, user interface input devices may include voice recognition sensing devices that enable users to interact with voice recognition systems (e.g., Siri® navigator), through voice commands.

[0065] User interface input devices may also include, without limitation, three dimensional (3D) mice, joysticks or pointing sticks, gamepads and graphic tablets, and audio/visual devices such as speakers, digital cameras, digital camcorders, portable media players, webcams, image scanners, fingerprint scanners, barcode reader 3D scanners, 3D printers, laser rangefinders, and eye gaze tracking devices. Additionally, user interface input devices may include, for example, medical imaging input devices such as computed tomography, magnetic resonance imaging, position emission tomography, medical ultrasonography devices. User interface input devices may also include, for example, audio input devices such as MIDI keyboards, digital musical instruments and the like.

[0066] User interface output devices may include a display subsystem, indicator lights, or non-visual displays such as audio output devices, etc. The display subsystem may be a Cathode Ray Tube (CRT), a flat-panel device, such as that using a Liquid Crystal Display (LCD) or plasma display, a projection device, a touch screen, and the like. In general, use of the term "output device" is intended to include all possible types of devices and mechanisms for outputting information from computer system 300 to a user or other computer. For example, user interface output devices may include, without limitation, a variety of display devices that visually convey text, graphics and audio/video information such as monitors, printers, speakers, headphones, automotive navigation systems, plotters, voice output devices, and modems.

[0067] Computer system 300 may comprise a storage subsystem 318 that comprises software elements, shown as being currently located within a system memory 310. System memory 310 may store program instructions that are loadable and executable on processing unit 304, as well as data generated during the execution of these programs.

[0068] Depending on the configuration and type of computer system 300, system memory 310 may be volatile (such as Random Access Memory (RAM)) and/or non-volatile (such as Read-Only Memory (ROM), flash memory, etc.) The RAM typically contains data and/or program modules that are immediately accessible to and/or presently being operated and executed by processing unit 304. In some cases, system memory 310 can comprise one or more Double Data Rate fourth generation (DDR4) Dual Inline Memory Modules (DIMMs). In some implementations, system memory 310 may include multiple different types of memory, such as Static Random Access Memory (SRAM) or Dynamic Random Access Memory (DRAM). In some implementations, a Basic Input/Output System (BIOS), containing the basic routines that help to transfer information between elements within computer system 300, such as during start-up, may typically be stored in the ROM. By way of example, and not limitation, system memory 310 also illustrates application programs 312, which may include client applications, Web browsers, mid-tier applications, Relational Database Management Systems (RDBMS), etc., program data 314, and an operating system 316. By way of example, operating system 316 may include various versions of Microsoft Windows®, Apple Macintosh®, and/or Linux operating systems, a variety of commercially-available UNIX® or UNIX-like operating systems (including without limitation the variety of GNU/Linux operating systems, the Google Chrome® OS, and the like) and/or mobile operating systems such as iOS, Windows® Phone, Android® OS, BlackBerry® 10 OS, and Palm® OS operating systems.

[0069] Storage subsystem 318 may also provide a tangible computer-readable storage medium for storing the basic programming and data constructs that provide the functionality of some embodiments. Software (programs, code modules, instructions) that when executed by a processor provide the functionality described above may be stored in storage subsystem 318. These software modules or instructions may be executed by processing unit 304. Storage

subsystem 318 may also provide a repository for storing data used in accordance with the present invention.

[0070] Storage subsystem 300 may also include a computer-readable storage media reader 320 that can further be connected to computer-readable storage media 322. Together and, optionally, in combination with system memory 310, computer-readable storage media 322 may comprehensively represent remote, local, fixed, and/or removable storage devices plus storage media for temporarily and/or more permanently containing, storing, transmitting, and retrieving computer-readable information.

[0071] Computer-readable storage media 322 containing code, or portions of code, can also include any appropriate media known or used in the art, including storage media and communication media, such as but not limited to, volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage and/or transmission of information. This can include tangible computer-readable storage media such as RAM, ROM, Electronically Erasable Programmable ROM (EEPROM), flash memory or other memory technology, CD-ROM, Digital Versatile Disk (DVD), or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or other tangible computer readable media. This can also include nontangible computer-readable media, such as data signals, data transmissions, or any other medium which can be used to transmit the desired information and which can be accessed by computing system 300.

[0072] By way of example, computer-readable storage media 322 may include a hard disk drive that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive that reads from or writes to a removable, nonvolatile magnetic disk, and an optical disk drive that reads from or writes to a removable, nonvolatile optical disk such as a CD ROM, DVD, and Blu-Ray® disk, or other optical media. Computer-readable storage media 322 may include, but is not limited to, Zip® drives, flash memory cards, Universal Serial Bus (USB) flash drives, Secure Digital (SD) cards, DVD disks, digital video tape, and the like. Computer-readable storage media 322 may also include, Solid-State Drives (SSD) based on non-volatile memory such as flash-memory based SSDs, enterprise flash drives, solid state ROM, and the like, SSDs based on volatile memory such as solid state RAM, dynamic RAM, static RAM,

DRAM-based SSDs, Magnetoresistive RAM (MRAM) SSDs, and hybrid SSDs that use a combination of DRAM and flash memory based SSDs. The disk drives and their associated computer-readable media may provide non-volatile storage of computer-readable instructions, data structures, program modules, and other data for computer system 300.

[0073] Communications subsystem 324 provides an interface to other computer systems and networks. Communications subsystem 324 serves as an interface for receiving data from and transmitting data to other systems from computer system 300. For example, communications subsystem 324 may enable computer system 300 to connect to one or more devices via the Internet. In some embodiments communications subsystem 324 can include Radio Frequency (RF) transceiver components for accessing wireless voice and/or data networks (e.g., using cellular telephone technology, advanced data network technology, such as 3G, 4G or Enhanced Data rates for Global Evolution (EDGE), WiFi (IEEE 802.11 family standards, or other mobile communication technologies, or any combination thereof), Global Positioning System (GPS) receiver components, and/or other components. In some embodiments communications subsystem 324 can provide wired network connectivity (e.g., Ethernet) in addition to or instead of a wireless interface. In some cases, communications subsystem 324 can be implemented in whole or in part as one or more PCIe cards.

[0074] In some embodiments, communications subsystem 324 may also receive input communication in the form of structured and/or unstructured data feeds 326, event streams 328, event updates 330, and the like on behalf of one or more users who may use computer system 300.

[0075] By way of example, communications subsystem 324 may be configured to receive data feeds 326 in real-time from users of social networks and/or other communication services such as Twitter® feeds, Facebook® updates, web feeds such as Rich Site Summary (RSS) feeds, and/or real-time updates from one or more third party information sources.

[0076] Additionally, communications subsystem 324 may also be configured to receive data in the form of continuous data streams, which may include event streams 328 of real-time events and/or event updates 330, that may be continuous or unbounded in nature with no explicit end.

Examples of applications that generate continuous data may include, for example, sensor data applications, financial tickers, network performance measuring tools (e.g. network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like.

[0077] Communications subsystem 324 may also be configured to output the structured and/or unstructured data feeds 326, event streams 328, event updates 330, and the like to one or more databases that may be in communication with one or more streaming data source computers coupled to computer system 300.

[0078] Computer system 300 can be one of various types, including a handheld portable device (e.g., an iPhone® cellular phone, an iPad® computing tablet, a PDA), a wearable device (e.g., a Google Glass® head mounted display), a PC, a workstation, a mainframe, a kiosk, a server rack, or any other data processing system.

[0079] Due to the ever-changing nature of computers and networks, the description of computer system 300 depicted in the figure is intended only as a specific example. Many other configurations having more or fewer components than the system depicted in the figure are possible. For example, customized hardware might also be used and/or particular elements might be implemented in hardware, firmware, software (including applets), or a combination. Further, connection to other computing devices, such as network input/output devices, may be employed. Based on the disclosure and teachings provided herein, a person of ordinary skill in the art will appreciate other ways and/or methods to implement the various embodiments.

[0080] As introduced above, embodiments of the invention provide systems and methods for managing processing, memory, storage, network, and cloud computing to significantly improve the efficiency and performance of processing nodes such as any of the servers or other computers or computing devices described above. Embodiments described herein can be implemented in a set of hardware components that, in essence, change the way in which processing, memory, storage, network, and cloud are managed by breaking down the artificial distinctions between processing, memory, storage and networking in today's commodity solutions to significantly improve the performance of commodity hardware. For example, the hardware elements can

include a standard format memory module, such as a Dual Inline Memory Module (DIMM), which can be added to any of the computer systems described above. For example, the memory module can be added to commodity or "off-the-shelf" hardware such a server node and acts as a big data accelerator within that node. The components can also include one or more object routers. Object routers can include, for example, a PCI express card added to the server node along with the memory module and one or more external object routers such as rack mounted routers, for example. Object routers can be used to interconnect two or more servers or other nodes adapted with the memory modules and help to manage processing, memory, and storage across these different servers Object routers can forward objects or portions of objects based on object addresses and participate in operation of the object memory fabric. Together, these hardware components can be used with commodity servers or other types of computing nodes in any combination to implement an object memory fabric architecture.

[0081] FIG. 4 is a block diagram illustrating an exemplary object memory fabric architecture according to one embodiment of the present invention. As illustrated here, the architecture 400 comprises an object memory fabric 405 supporting any number of applications 410a-g. As will be described in greater detail below, this object memory fabric 405 can comprise any number of processing nodes such as one or more servers having installed one or more memory modules as described herein. These nodes can be interconnected by one or more internal and/or external object routers as described herein. While described as comprising one or more servers, it should be noted that the processing nodes of the object memory fabric 405 can comprise any of a variety of different computers and/or computing devices adapted to operate within the object memory fabric 405 as described herein.

[0082] According to one embodiment, the object memory fabric 405 provides an object-based memory which manages memory objects within the memory of the nodes of the object memory fabric 405 and at the memory layer rather than in the application layer. That is, the objects and associated properties can be implemented and managed natively in the nodes of the object memory fabric 405 to provide increased functionality without any software and increasing efficiency and performance by dynamically managing object characteristics including, but not limited to persistence, location and processing. Object properties can also propagate to the

applications 410a-g. The memory objects of the object memory fabric 405 can be used to eliminate typical size constraints on memory space of the commodity servers or other nodes imposed by address sizes. Rather, physical addressing can be managed within the memory objects themselves and the objects can in turn be accessed and managed through the object name space. The memory objects of the object memory fabric 405 can also be used to eliminate the distinction between memory (temporary) and storage (persistent) by implementing and managing both within the objects. The object memory fabric 405 can also eliminate the distinction between local and remote memory by transparently managing the location of objects (or portions of objects) so all objects appear simultaneously local to all nodes. The memory objects can also eliminate the distinction between processing and memory through methods of the objects to place the processing within the memory itself. In other words, embodiments of the present invention provide a single-level memory that puts the computes with the storage and the storage with the computes, directly and thereby eliminating numerous levels of software overhead communicating across these levels and the artificial overhead of moving data to be processed.

[0083] In these ways, embodiments of the object memory fabric 405 and components thereof as described herein can provide transparent and dynamic performance acceleration, especially with big data or other memory intensive applications by reducing or eliminating overhead typically associated with memory management, storage management, networking, data directories, and data buffers at both the system and application software layers. Rather, management of the memory objects at the memory level can significantly shorten the pathways between storage and memory and between memory and processing, thereby eliminating the associated overhead between each.

[0084] Embodiments provide coherent, hardware-based, infinite memory managed as memory objects with performance accelerated in-memory, spanning all nodes, and scalable across all nodes. This enables transparent dynamic performance acceleration based on the object and end application. Using an architecture according to embodiments of the present invention, applications and system software can be treated the same and as simple as a single, standard server but additionally allowing memory fabric objects to capture heuristics. Embodiments provide multiple dimensions of accelerated performance including locality acceleration.

According to one embodiment, object memory fabric metadata associated with the memory objects can include triggers which enable the object memory fabric architecture to localize and move data to fast dram memory ahead of use. Triggers can be a fundamental generalization that enables the memory system to execute arbitrary functions based on memory access. Various embodiments can also include an instruction set which can provide a unique instruction model for the object memory fabric based on the triggers defined in the metadata associated with each memory object and that supports core operations and optimizations and allows the memory intensive portion of applications to be more efficiently executed in a highly parallel manner within IMF.

[0085] Embodiments can also decrease software path-length by substituting a small number of memory references for a complex application, storage and network stack. This can be accomplished when memory and storage is directly addressable as memory under embodiments of the present invention. Embodiments can additionally provide accelerated performance of high level memory operations. For many cases, embodiments of the object memory fabric architecture can eliminate the need to move data to the processor and back to memory, which is extremely inefficient for today's modern processors with three or more levels of caches.

[0086] FIG. 5 is a block diagram illustrating an exemplary memory fabric object memory according to one embodiment of the present invention. More specifically, this example illustrates an application view of how memory fabric object memory can be organized. Memory fabric object address space 500 can be a 128 bit linear address space where the object ID corresponds to the start of the addressable object. Objects 510 can be variable size from 2^{12} to 2^{64} bytes. The address space 500 can efficiently be utilized sparsely within and across objects as object storage is allocated on a per block basis. The size of the object space 500 is meant to be large enough that garbage collection is not necessary and to enable disjoint systems to be easily combined.

[0087] Object metadata 505 associated with each object 510 can be transparent with respect to the object address space 500 and can utilize the object memory fabric to manage objects and blocks within objects and can be accessible at appropriate privilege by applications 515a-g through Application Program Interfaces (APIs) of the object memory fabric. This API provides

functions for applications to set up and maintain the object memory fabric, for example by using modified Linux libc. With a small amount of additional effort applications such as a SQL database or graph database can utilize the API to create memory objects and provide and/or augment object metadata to allow the object memory fabric to better manage objects. Object metadata 505 can include object methods, which enable performance optimization through dynamic object-based processing, distribution, and parallelization. Metadata can enable each object to have a definable security policy and access encapsulation within an object.

[0088] According to embodiments of the present invention, applications 515a-g can now access a single object that captures it's working and/or persistent data (such as App0 515a) or multiple objects for finer granularity (such as App1 515b). Applications can also share objects. Object memory 500 according to these embodiments can physically achieves this powerfully simple application view with a combination of physical organization, which will be described in greater detail below with reference to FIG. 6, and object memory dynamics. Generally speaking, the object memory 500 can be organized as a distributed hierarchy that creates hierarchical neighborhoods for object storage and applications 515a-g. Object memory dynamics interact and leverage the hierarchal organization to dynamically create locals of objects and applications (object methods) that operate on objects. Since object methods can be associated with memory objects, as objects migrate and replicate on the memory fabric, object methods naturally gain increased parallelism as object size warrants. The hierarchy in conjunction with object dynamics can further create neighborhoods of neighborhoods based on the size and dynamics of the object methods.

[0089] FIG. 6 is a block diagram illustrating an exemplary object memory dynamics and physical organization according to one embodiment of the present invention. As illustrated in this example, an object memory fabric 600 as described above can include any number of processing nodes 605 and 610 communicatively coupled via one or more external object routers 615. Each node 605 and 610 can also include an internal object router 620 and one or more memory modules. Each memory module 625 can include a node object memory 635 supporting any number of applications 515a-g. Generally speaking, the memory module 625, node object router 620 and inter-node object router 615 can all share a common functionality with respect to

the object memory 635 and index thereof. In other words, the underlying design objects can be reused in all three providing a common design adaptable to hardware of any of a variety of different form factors and types in addition to those implementations described here by way of example.

[0090] More specifically, a node can comprise a single node object router 620 and one or more memory modules 625 and 630. According to one embodiment, a node 605 can comprise a commodity or "off-the-shelf" server, the memory module 625 can comprise a standard format memory card such as a Dual-Inline Memory Module (DIMM) card, and the node object router 620 can similarly comprise a standard format card such as a Peripheral Component Interconnect express (PCIe) card. The node object router 620 can implement an object index covering the objects/blocks held within the object memory(s) 635 of the memory modules 625 and 630 within the same node 605. Each memory module 625 and 630 can hold the actual objects and blocks within objects, corresponding object meta-data, and object index covering objects currently stored local to that memory module. Each memory module 625 and 630 can independently manage both dram memory (fast and relatively expensive) and flash memory (not as fast, but much less expensive) in a manner that the processor (not shown) of the node 605 thinks that there is the flash amount of fast dram. The memory modules 625 and 630 and the node object router 620 can both manage free storage through a free storage index implemented in the same manner as for other indexes. Memory modules 625 and 630 can be directly accessed over the standard DDR memory bus by processor caches and processor memory reference instructions. In this way, the memory objects of the memory modules 625 and 630 can be accessed using only conventional memory reference instructions and without implicit or explicit Input/Output (I/O) instructions.

[0091] Objects within the object memory 635 of each node 625 can be created and maintained through an object memory fabric API (not shown). The node object router 620 can communicate with the API through a modified object memory fabric version of libc and an object memory fabric driver (not shown). The node object router 620 can then update a local object index, send commands toward a root, i.e., towards the inter-node object router 615, as required and communicate with the appropriate memory module 625 or 630 to complete the API command

locally. The memory module 625 or 630 can communicate administrative requests back to the node object router 620 which can handle them appropriately.

[0092] According to one embodiment, the internal architecture of the node object router 620 can be very similar to the memory module 625 with the differences related to routing functionality such as managing a node memory object index and routing appropriate packets to and from the memory moduels 625 and 630 and the inter-node object router 615. That is, the node object router 620 can have additional routing functionality but does not need to actually store memory objects.

[0093] The inter-node object router 615 can be considered analogous to an IP router. However, the first difference is the addressing model used. IP routers utilize a fixed static address per each node and routes based on the destination IP address to a fixed physical node. However, the inter-node object router 615 of the object memory fabric 600 utilizes a memory fabric object address (OA) which specifies the object and specific block of the object. Objects and blocks can dynamically reside at any node. The inter-node object router 615 can route OA packages based on the dynamic location(s) of objects and blocks and track object/block location dynamically in real time. The second difference is that the object router can implement the object memory fabric distributed protocol which provides the dynamic nature of object/block location and object functions, for example including, but not limited, to triggers. The inter-node object router 615 can be implemented as a scaled up version of node object router 620 with increased object index storage capacity, processing rate and overall routing bandwidth. Also, instead of connecting to a single PCIe or other bus or channel to connect to memory modules, inter-node object router 615 can connect to multiple node object routers and/or multiple other inter-node object routers. According to one embodiment, a node object router 620 can communicate with the memory modules 625 and 630 with direct memory access over PCIe and the memory bus (not shown) of the node 605. Node object routers of different nodes 605 and 610 can in turn connect with one or more inter-node object routers 615 over a high-speed network (not shown) such as 25/100GE fiber that uses several layers of Gigabit Ethernet protocol or object memory fabric protocol tunneled through standard IP, for example. Multiple inter-node object routers can connect with the same network.

[0094] In operation, the memory fabric object memory can physically achieve its powerfully simple application view described above with reference to FIGs. 4 and 5 with a combination of physical organization and object memory dynamics. According to one embodiment and as introduced above with reference to FIG. 5, the memory fabric object memory can be organized as a distributed hierarchy that creates hierarchical neighborhoods for object storage and applications 515a-g. The node object routers can keep track of which objects and portions of objects are local to a neighborhood. The actual object memory can be located on nodes 605 or 610 close to applications 515a-g and memory fabric object methods.

[0095] Also as introduced above, object memory dynamics can interact and leverage the hierarchal organization to dynamically create locals of objects and applications (object methods) that operate on objects. Since object methods can be associated with objects as objects migrate and replicate across nodes, object methods naturally gain increased parallelism as object size warrants. This object hierarchy, in conjunction with object dynamics, can in turn create neighborhoods of neighborhoods based on the size and dynamics of the object methods.

[0096] For example, App0 515a spans multiple memory modules 625 and 630 within a single level object memory fabric neighborhood, in this case node 605. Object movement can stay within that neighborhood and its node object router 620 without requiring any other communication links or routers. The self-organizing nature along the hierarchy defined neighborhoods provides efficiency from a performance and minimum bandwidth perspective. In another example, App1 (A1) 515b can have the same characteristic but in a different neighborhood, i.e., in node 610. App2 (A2) 515c can be a parallel application across a two-level hierarchy neighborhood, i.e., nodes 605 and 610. Interactions can be self-contained in the respective neighborhood.

[0097] In the foregoing description, for the purposes of illustration, methods were described in a particular order. It should be appreciated that in alternate embodiments, the methods may be performed in a different order than that described. It should also be appreciated that the methods described above may be performed by hardware components or may be embodied in sequences of machine-executable instructions, which may be used to cause a machine, such as a general-purpose or special-purpose processor or logic circuits programmed with the instructions to

perform the methods. These machine-executable instructions may be stored on one or more machine readable mediums, such as CD-ROMs or other type of optical disks, floppy diskettes, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, flash memory, or other types of machine-readable mediums suitable for storing electronic instructions. Alternatively, the methods may be performed by a combination of hardware and software.

[0098] While illustrative and presently preferred embodiments of the invention have been described in detail herein, it is to be understood that the inventive concepts may be otherwise variously embodied and employed, and that the appended claims are intended to be construed to include such variations, except as limited by the prior art.

WHAT IS CLAIMED IS:

1. A hardware-based processing node of an object memory fabric, the processing node comprising:

a memory module storing and managing one or more memory objects, wherein:

each memory object is created natively within the memory module,

each memory object is accessed using a single memory reference instruction without Input/Output (I/O) instructions,

each memory object is managed by the memory module at a single memory layer, and

physical addressing of both memory and storage of the object memory fabric is managed with each of the one or more memory objects through an object name space of the object memory fabric.

2. The hardware-based processing node of claim 1, wherein the memory module manages both storage and memory without distinction through the one or more memory objects.

3. The hardware-based processing node of claim 2, wherein managing both storage and memory without distinction through the one or more memory objects comprises managing all of the one or more memory objects as memory regardless of an underlying physical storage media.

4. The hardware-based processing node of claim 1, wherein the object memory fabric comprises a plurality of hardware-based processing nodes and wherein the one or more memory objects are accessed and managed across the object memory fabric through the object name space of the object memory fabric.

5. The hardware-based processing node of claim 4, wherein each memory object and properties of each memory object are maintained on any one or more of the plurality of nodes in the object memory fabric, wherein managing the memory objects includes

maintaining the memory objects and properties of the memory objects as the memory objects are moved, split, or duplicated between nodes

6. The hardware-based processing node of claim 1, wherein the object name space is unconstrained by the physical addresses managed by the one or more memory objects.

7. The hardware-based processing node of claim 1, wherein the hardware-based processing node comprises a Dual In-line Memory Module (DIMM) card.

8. The hardware-based processing node of claim 1, wherein the hardware-based processing node comprises a commodity server and wherein the memory module comprises a Dual In-line Memory Module (DIMM) card installed within the commodity server.

9. The hardware-based processing node of claim 8, further comprising a communication interface coupled with the object memory fabric.

10. The hardware-based processing node of claim 9, wherein the communication interface comprises a Peripheral Component Interconnect Express (PCI-e) card.

11. The hardware-based processing node of claim 1, wherein the hardware-based processing node comprises a mobile computing device.

12. The hardware-based processing node of claim 1, wherein the hardware-based processing node comprises a single chip.

13. An object memory fabric comprising:

a plurality of hardware-based processing nodes, each hardware-based processing node comprising:

one or more memory modules storing and managing one or more memory objects, wherein each memory object is created natively within the memory module, each memory object is accessed using a single memory reference instruction without Input/Output (I/O) instructions, each memory object

is managed by the memory module at a single memory layer, and physical addressing of both memory and storage of the object memory fabric is managed with each of the one or more memory objects through an object name space of the object memory fabric, and

a node router communicatively coupled with each of the one or more memory modules of the node and adapted to route memory objects or portions of memory objects between the one or more memory modules of the node; and

one or more inter-node routers communicatively coupled with each node router, wherein each of the plurality of nodes of the object memory fabric is communicatively coupled with at least one of the inter-node routers and adapted to route memory objects or portions of memory objects between the plurality of nodes.

14. The object memory fabric of claim 13, wherein the memory module manages both storage and memory without distinction through the one or more memory objects.

15. The object memory fabric of claim 14, wherein managing both storage and memory without distinction through the one or more memory objects comprises managing all of the one or more memory objects as memory regardless of an underlying physical storage media.

16. The object memory fabric of claim 13, wherein the one or more memory objects are accessed and managed across the object memory fabric through the object name space of the object memory fabric.

17. The object memory fabric of claim 16, wherein each memory object and properties of each memory object are maintained on any one or more of the plurality of nodes in the object memory fabric, wherein managing the memory objects includes maintaining the memory objects and properties of the memory objects as the memory objects are moved, split, or duplicated between nodes

18. The object memory fabric of claim 13, wherein the object name space is unconstrained by the physical addresses managed by the one or more memory objects.

19. The object memory fabric of claim 12, wherein at least one hardware-based processing node comprises a commodity server and wherein the one or more memory modules of the commodity server comprise at least one Dual In-line Memory Module (DIMM) card installed within the commodity server.

20. The object memory fabric of claim 13, wherein the communication interface comprises a Peripheral Component Interconnect Express (PCI-e) card.

21. The object memory fabric of claim 13, wherein at least one hardware-based processing node comprises a mobile computing device.

22. The object memory fabric of claim 13, wherein at least one hardware-based processing node comprises a single chip.

23. A method for storing and managing one or more memory objects in an object memory fabric, the method comprising:

creating each memory object natively within a memory module of a hardware-based processing node of the object memory fabric;

accessing each memory object using a single memory reference instruction without Input/Output (I/O) instructions;

managing each memory object within the memory module at a single memory layer; and

managing physical addressing of both memory and storage of the object memory fabric with each of the one or more memory objects through an object name space of the object memory fabric.

24. The method of claim 23, wherein the memory module manages both storage and memory without distinction through the one or more memory objects.

25. The method of claim 24, wherein managing both storage and memory without distinction through the one or more memory objects comprises managing all of the one or more memory objects as memory regardless of an underlying physical storage media.

26. The method of claim 23, wherein the object memory fabric comprises a plurality of hardware-based processing nodes and wherein the one or more memory objects are accessed and managed across the object memory fabric through the object name space of the object memory fabric.

27. The method of claim 26, wherein each memory object and properties of each memory object are maintained on any one or more of the plurality of nodes in the object memory fabric, wherein managing the memory objects includes maintaining the memory objects and properties of the memory objects as the memory objects are moved, split, or duplicated between nodes

28. The method of claim 23, wherein the object name space is unconstrained by the physical addresses managed by the one or more memory objects.

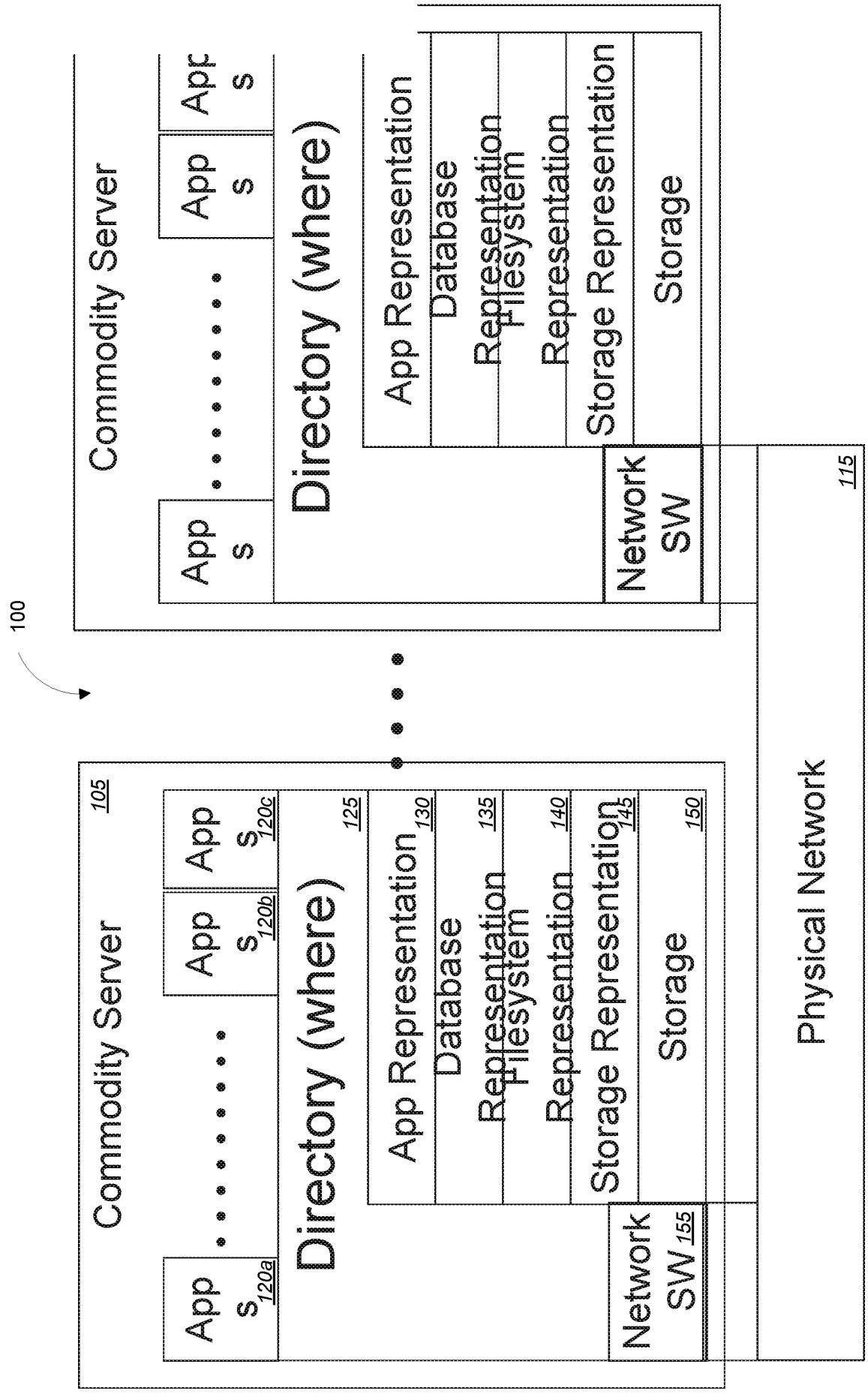

FIG. 1
(Prior Art)

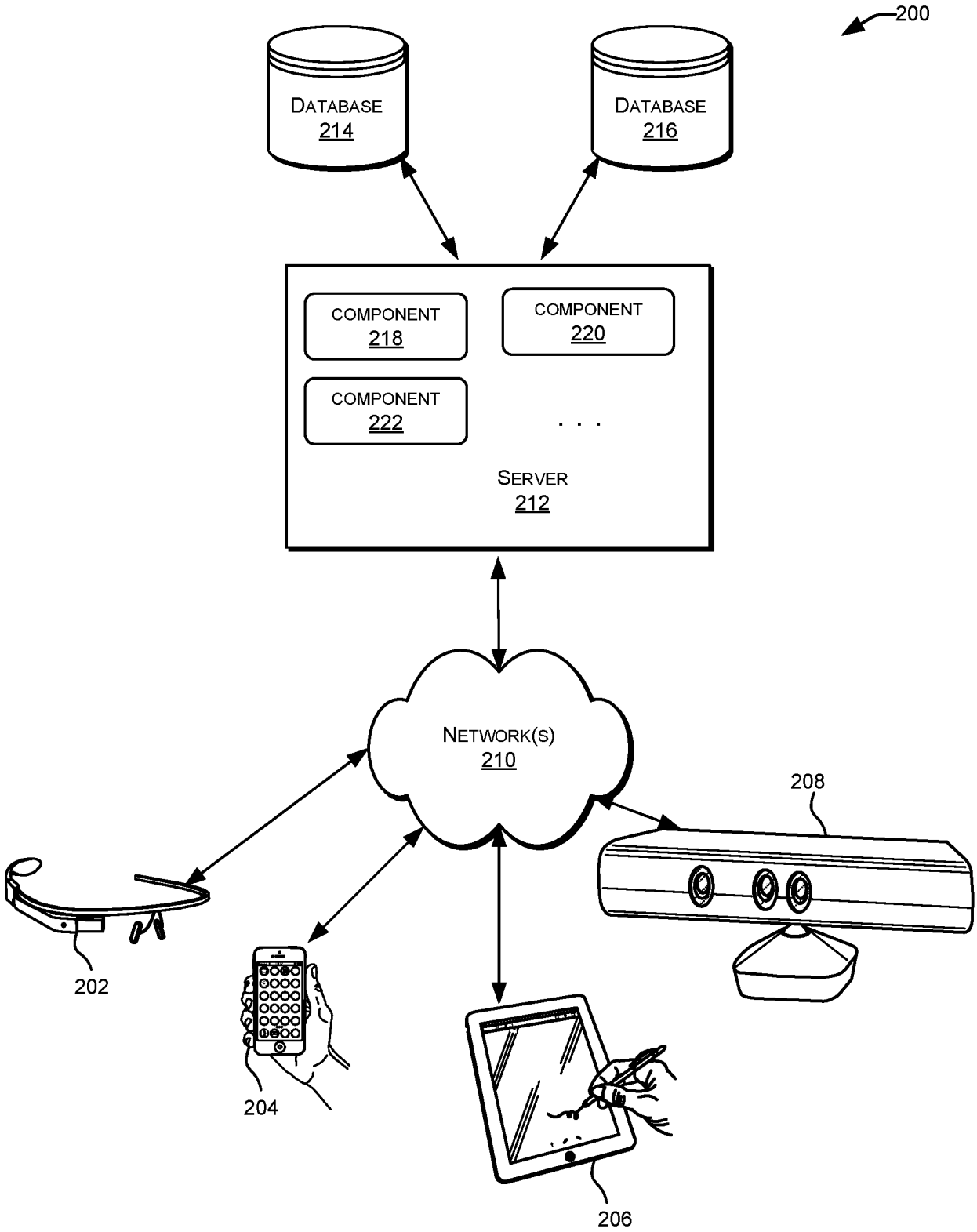
200
DATABASE
214
DATABASE
216
COMPONENT
218
COMPONENT
220
COMPONENT
222
. . .
SERVER
212
NETWORK(S)
210
208
202
204
206

FIG. 2

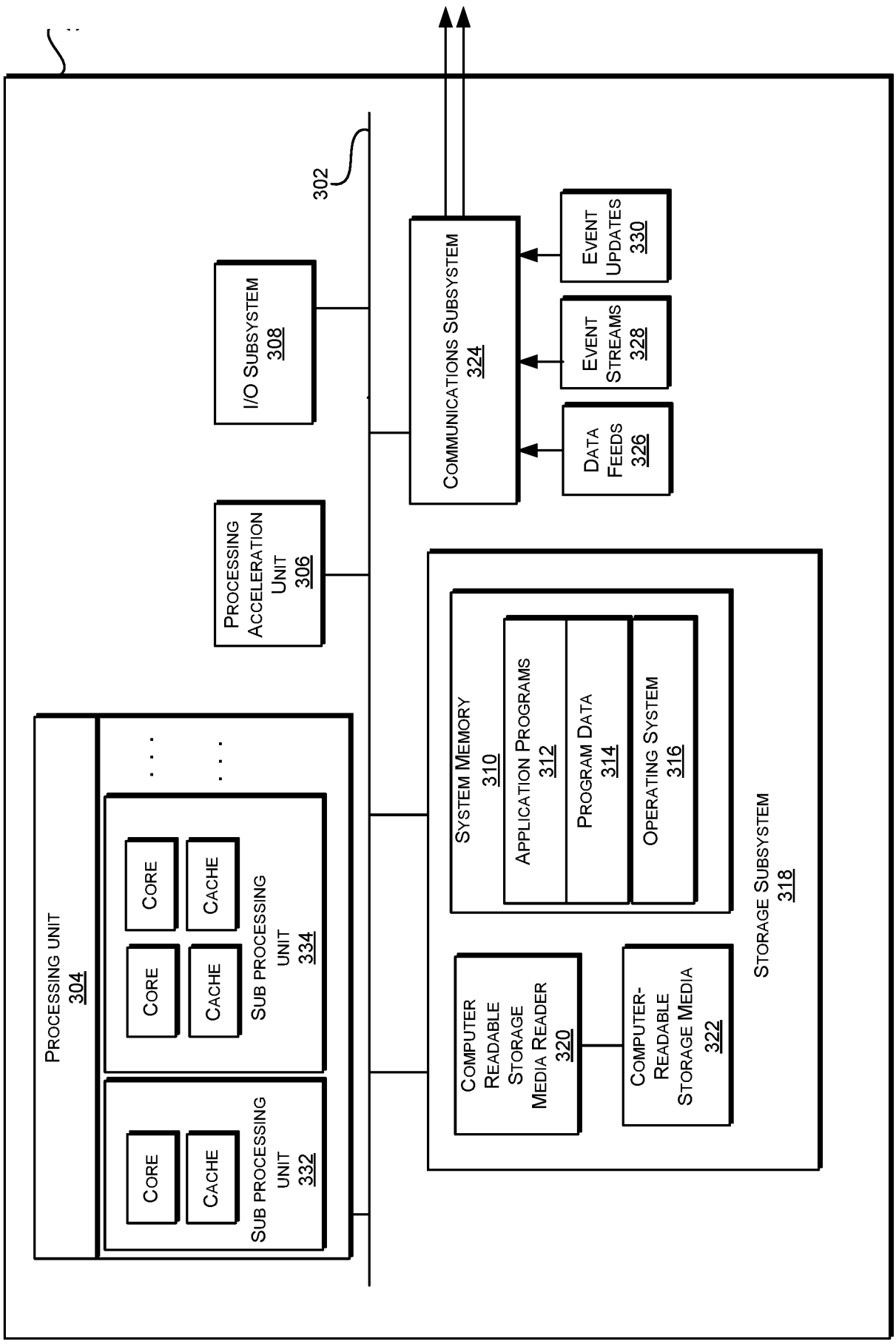
302
I/O SUBSYSTEM 308
COMMUNICATIONS SUBSYSTEM 324
EVENT UPDATES 330
EVENT STREAMS 328
DATA FEEDS 326
PROCESSING ACCELERATION UNIT 306
PROCESSING UNIT 304
CORE
CACHE
SUB PROCESSING UNIT 332
SUB PROCESSING UNIT 334
SYSTEM MEMORY 310
APPLICATION PROGRAMS 312
PROGRAM DATA 314
OPERATING SYSTEM 316
COMPUTER READABLE STORAGE MEDIA READER 320
COMPUTER-READABLE STORAGE MEDIA 322
STORAGE SUBSYSTEM 318

FIG. 3

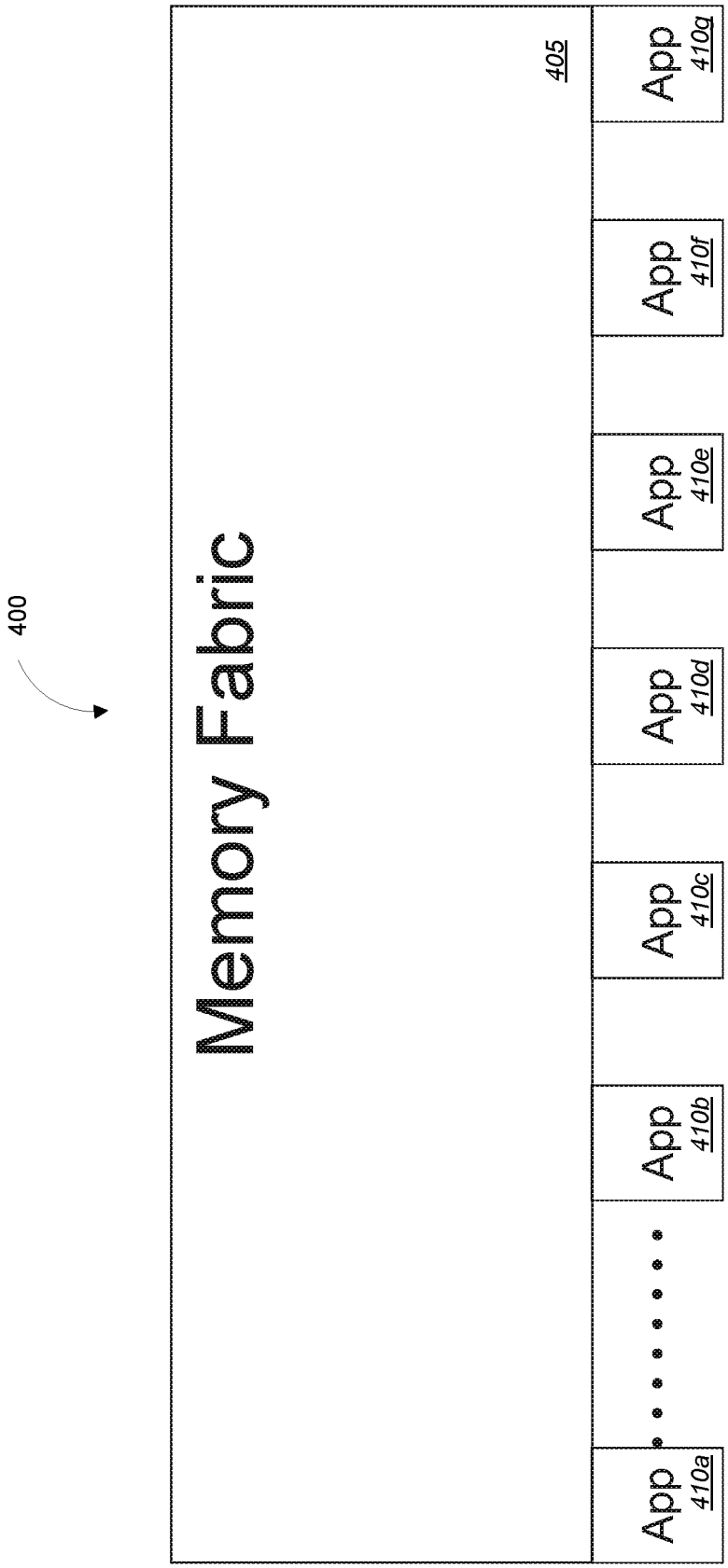
400
Memory Fabric
405
App 410a
App 410b
App 410c
App 410d
App 410e
App 410f
App 410g

FIG. 4

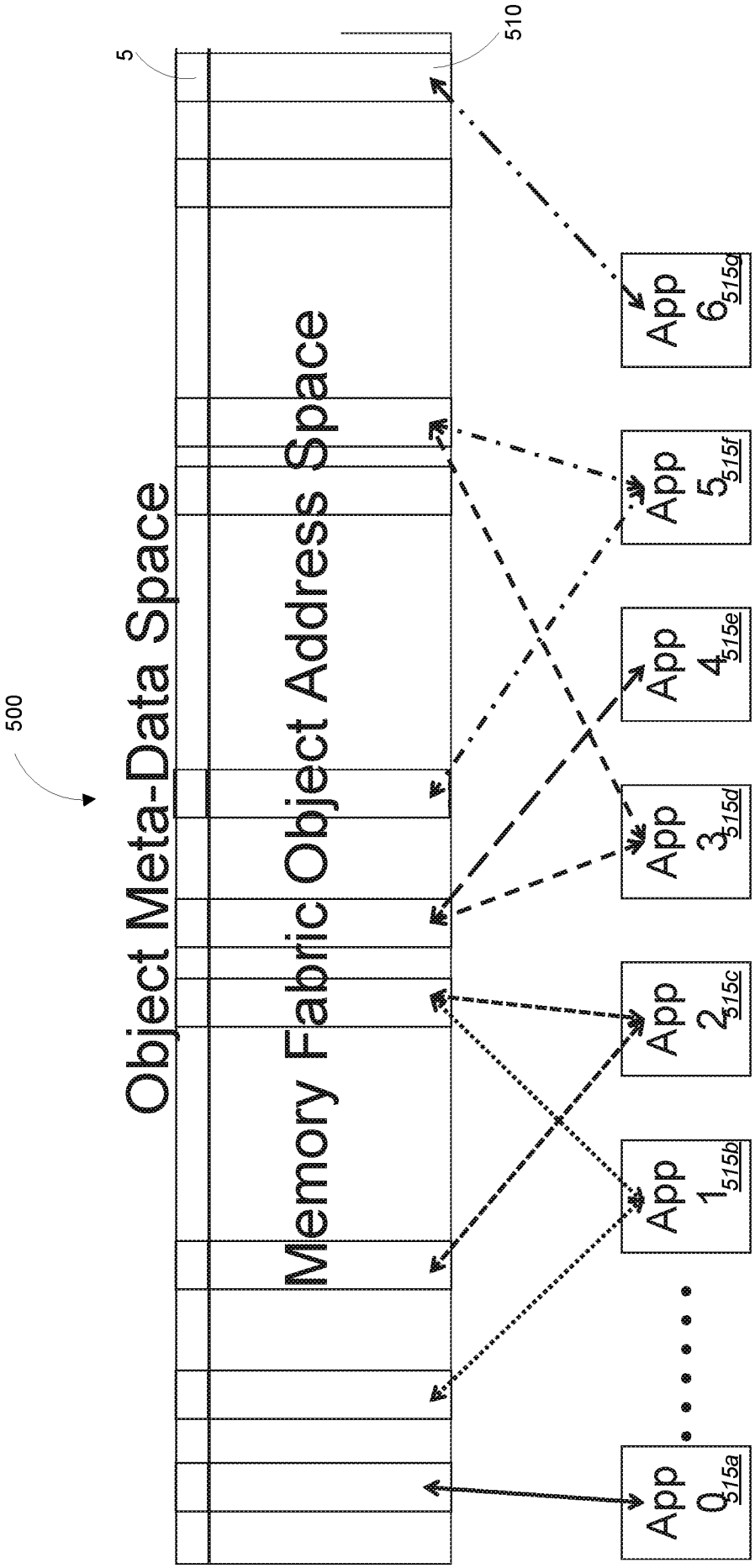
500
5
510
Object Meta-Data Space
Memory Fabric Object Address Space
App 6 515g
App 5 515f
App 4 515e
App 3 515d
App 2 515c
App 1 515b
App Q 515a

FIG. 5

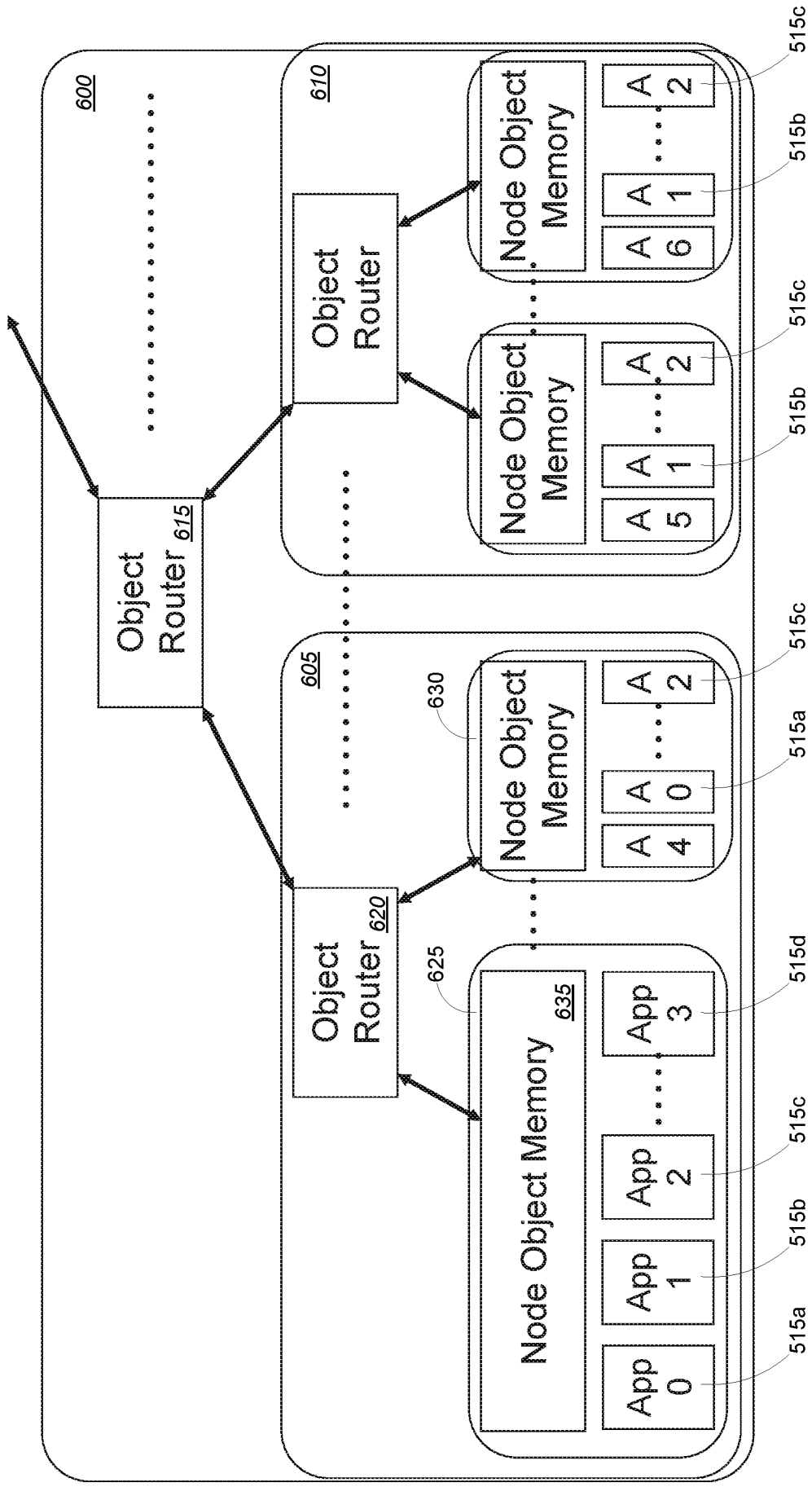
600
Object Router 615
610
Object Router
Node Object Memory
A 6
A 1
A 2
A 5
A 1
A 2
515c
515b
515c
515b
605
630
Object Router 620
Node Object Memory
A 4
A 0
A 2
515c
515a
625
635
Node Object Memory
App 0
App 1
App 2
App 3
515d
515c
515b
515a

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 16/14024

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 13/00 (2016.01)
CPC - G06F13/18, G06F13/1642, G06F13/1605, G06F9/52, G06F15/167, G06F3/067, G06F3/0605, G06F3/0659, G06F3/0613
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 13/00 (2016.01); CPC: G06F13/18, G06F13/1642, G06F13/1605, G06F9/52, G06F15/167, G06F3/067, G06F3/0605, G06F3/0659, G06F3/0613, G06F3/0689

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 711/151, 711/154, 709/224, 709/218, 709/229, 709/201, 709/217, 709/223, 709/219, 709/226

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PatBase, ProQuest Dialog, Google Web, Google Patents (Search terms: memory object, buffer, cache, data object, commodity server, computer, processor, DIMM, interconnect express, PCI, PCIE, native, create, node router, switch, memory fabric, memory reference instruction, address, memory layer, mobile commodity computer, mobile processing node, etc.)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 2005/0273571 A1 (Lyon et al.) 08 December 2005 (08.12.2005), para. [0023]-[0024], [0029], [0031], [0038], [0041], [0046], and [0074], and Figs. 2 and 8, and claims 1 and 13.	1-6, 13-18, 23-28 --- 7-12, 19-22
Y	US 2014/0165196 A1 (Dalal et al.) 12 June 2014 (12.06.2014), para. [0024]-[0025], [0029], [0034], [0036], and [0102]-[0103], and Fig. 1.	7-10, 19-20
Y	US 5,664,207 A (Crumpler et al.) 02 September 1997 (02.09.1997), col. 6, ln. 22-34 and 56-57, and Figs. 1-2.	11, 21
Y	US 2014/0317352 A1 (Kleen) 23 October 2014 (23.10.2014), para. [0015], [0026], and [0028].	12, 22
A	US 2012/0017037 A1 (Riddle et al.) 19 January 2012 (19.01.2012), entire document.	1-28

☐ 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
05 March 2016 (05.03.2016)	28 MAR 2016
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. 571-273-8300	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (January 2015)