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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 June 2017 (29.06.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/112346 A1

(51) International Patent Classification:
*G06F 13/40* (2006.01)

(21) International Application Number:
PCT/US2016/063795

(22) International Filing Date:
26 November 2016 (26.11.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/757,892 24 December 2015 (24.12.2015) US

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054-1549 (US).

(72) Inventors: **DINAN, James**; 75 Reed Road, Hudson, MA 01749 (US). **FLAJSLIK, Mario**; 34 Pond St., Hopkinton, Massachusetts 01748 (US). **UNDERWOOD, Keith**; 705 East Copeland Dr., Powell, Tennessee 37849 (US). **KEPPEL, David**; 1011 Dale Ave., Moutain View, California 94040 (US). **HANEBUTTE, Ulf Rainer**; 11006 56th St. NW, Gig Harbor, Washington 98335 (US).

(74) Agent: **CRANDALL, Sean C.**; Patent Capital Group, 2816 Lago Vista Lane, Rockwall, Texas 75032 (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))*

*[Continued on next page]*

(54) Title: FABRIC-INTEGRATED DATA PULLING ENGINE

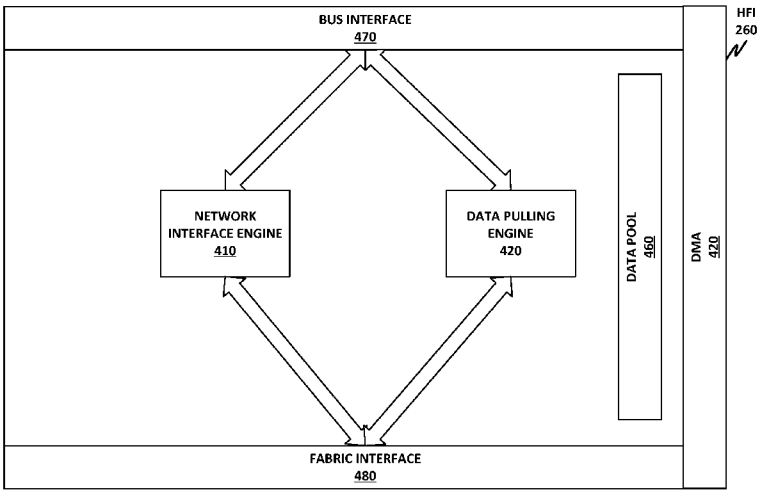


*Fig. 4*

(57) Abstract: In an example, there is disclosed a compute node, comprising: first one or more logic elements comprising a data producer engine to produce a datum; and a host fabric interface to communicatively couple the compute node to a fabric, the host fabric interface comprising second one or more logic elements comprising a data pulling engine, the data pulling engine to: publish the datum as available; receive a pull request for the datum, the pull request comprising a node identifier for a data consumer; and send the datum to the data consumer via the fabric. There is also disclosed a method of providing a data pulling engine.

WO 2017/112346 A1

**Published**:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

# FABRIC-INTEGRATED DATA PULLING ENGINE

## Cross-Reference to Related Application

**[0001]** This application claims the benefit of priority to U.S. Nonprovisional Patent Application No. 14/757,892 filed 24 December 2015 entitled, "FABRIC-INTEGRATED DATA PULLING ENGINE", which is incorporated by reference herein in its entirety.

## Statement Regarding Federally-Funded Research

**[0002]** This invention was made with Government support under contract number H98230-13-D-0124 awarded by the Department of Defense. The Government has certain rights in this invention

## Field of the specification

**[0003]** This disclosure relates in general to the field of high-performance computing, and more particularly, though not exclusively to, a system and method for providing a fabric-integrated data pulling engine.

## Background

**[0004]** High-performance computing, also called cluster computing, is a computing strategy in which a large number of processing cores are tightly coupled so that they can perform large computations in parallel. Data to be operated on may be divided into a number of slices that can be distributed across many different cores. A large number of cores may simultaneously perform the same operation on different data, and then report the result. In this context, a "producer" is a node that has data available. A "consumer" is a node that is to receive those data.

## Brief Description of the Drawings

**[0005]** The present disclosure is best understood from the following detailed description when read with the accompanying figures. It is emphasized that, in accordance with the standard practice in the industry, various features are not necessarily drawn to scale, and are used for illustration purposes only. Where a scale is shown, explicitly or implicitly, it provides only one illustrative example. In other embodiments, the dimensions of the various features may be arbitrarily increased or reduced for clarity of discussion.

**[0006]** FIGURE 1 is a block diagram of a high-performance computing (HPC) system according to one or more examples of the present specification.

**[0007]** FIGURE 2 is a block diagram of a compute node according to one or more examples of the present specification.

**[0008]** FIGURE 3 is a block diagram of a producer-consumer architecture according to one or more examples of the present specification.

**[0009]** FIGURE 4 is a block diagram of a host fabric interface (HFI) according to one or more examples of the present specification.

**[0010]** FIGURE 5 is a block diagram of species of memory according to one or more examples of the present specification.

**[0011]** FIGURE 6 is a flow chart of a method performed by a data producer according to one or more examples of the present specification.

**[0012]** FIGURE 7 is a flow chart of a method performed by an HFI data pulling engine (DPE) according to one or more examples of the present specification.

**[0013]** FIGURE 8 is a flow chart of a method performed by a data consumer according to one or more examples of the present specification.

## Summary

**[0014]** In an example, there is disclosed a compute node, comprising: first one or more logic elements comprising a data producer engine to produce a datum; and a host fabric interface to communicatively couple the compute node to a fabric, the fabric interface comprising second one or more logic elements comprising a data pulling engine, the data pulling engine to: publish the datum as available; receive a pull request for the datum, the pull request comprising a node identifier for a data consumer; and send the datum to the data consumer via the fabric. There is also disclosed a method of providing a data pulling engine.

## Embodiments of the Disclosure

**[0015]** The following disclosure provides many different embodiments, or examples, for implementing different features of the present disclosure. Specific examples of components and arrangements are described below to simplify the present disclosure. These are, of course, merely examples and are not intended to be limiting. Further, the present disclosure may repeat reference numerals and/or letters in the various examples. This repetition is for the purpose of

simplicity and clarity and does not in itself dictate a relationship between the various embodiments and/or configurations discussed. Different embodiments may have different advantages, and no particular advantage is necessarily required of any embodiment.

**[0016]** Some existing HPC fabrics and communication stacks face limitations in providing efficient support for data transfers in which producers and consumers are paired dynamically. In such relationships, a producer may not know the identity of the consumer. For example, in dynamic load balancing using work stealing, or in a thread pool programming pattern such as a web server, the producers of work or data may not know who the consumer will be. As a result, some existing architectures require multiple communication operations to to coordinate a dynamic producer-consumer data transfer.

**[0017]** For example, consider the implementation of a dynamic data transfer using the Message Passing Interface (MPI), which is an industry-standard HPC communication library that provides two-sided (i.e., matched send and receive) messaging. MPI assumes a protocol where senders initiate contact with receivers; as a result, it does not provide direct support for an anonymous send operation. Instead, dynamic data transfers in MPI's matched send-receive model require an application-level request-response protocol to be used, in which the producer periodically checks for incoming requests from any consumer, and generates a response once it sees the request.

**[0018]** Certain embodiments of this approach incurs overheads that can severely impact performance. For example:

a. The producer incurs a polling overhead.
b. The producer's polling interval generates latency to the consumer because of the resulting delay in responding to incoming requests.
c. Additional coordination messages must be transmitted generating both fabric and endpoint overheads.

**[0019]** On the other hand, one-sided communication models also exist, such as "symmetric hierarchical memory" (OpenSHMEM), and the MPI remote memory access (RMA) interface. These may provide asynchronous read (get), write (put), and atomic update operations. These operations can eliminate polling and latency overheads by allowing the consumer to directly access the producer's memory. However, additional synchronization between all of the consumers may need to be performed to locate and match available data entries, and to ensure that two consumers don't corrupt memory when accessing the producer's

memory directly. This synchronization impacts communication efficiency and can incur serialization overheads when multiple consumers request data from the same producer.

**[0020]** The present specification describes, in one embodiment, a communication operation comprising a one-sided pull, which allows consumers to request data from a producer using a single, asynchronous communication operation. In an example, this communication model is supported efficiently through a Data Pulling Engine (DPE) that is integrated with the fabric endpoint, and coordinates consumption of a producer's data by multiple consumers.

**[0021]** This one-sided pull is suitable for implementing dynamic producer-consumer data transfers. This mechanism can be efficiently offloaded to suitable hardware, such as in one embodiment hardware supporting the "Portals 4" interface.

**[0022]** In an embodiment, the Data Pulling Engine (DPE) is integrated with a fabric endpoint and eliminates the overheads described in the previous embodiments. In particular, the engine is immediately responsive when a pull is received, eliminating polling overheads incurred by both the producer and consumer. The engine is able to achieve a pull with a single round-trip message, initiated by the consumer, increasing communication efficiency further. Because the HFI-integrated DPE manages responding to multiple consumers, it also minimizes serialization overheads.

**[0023]** A one-sided pull is an operation that can be used by a consumer to retrieve information from a producer. The DPE of the present specification differs from two-sided messaging in that the producer application is passive and does not participate directly in the data transfer. It also differs from one-sided messaging (e.g. an OpenSHMEM or MPI one-sided get operation) in that the consumer does not select a specific datum it will retrieve. Rather, it requests a datum from a data species, and can receive any datum from that species. A further distinction from one-sided get is that once a datum has been pulled, it is ensured that the same datum will not be retrieved by a subsequent pull operation.

**[0024]** The present DPE can be implemented using hardware and software components, and may be integrated with a host fabric interface (HFI) in an HPC system to achieve the greatest benefit. The DPE intercepts one-sided pull operations performed by consumers, matches them against data that have been produced, and provides an immediate and asynchronous response.

**[0025]** In an embodiment, the DPE maintains a data pool that contains data elements available to consumers. Data in the pool may be produced locally or remotely, and are added to the DPE at a particular node in an HPC system through a locally or remotely performed push

operation. Data tracked by the pool may have an application-supplied tag, indicating a data species. Items in the same species may have an associated ordering (e.g., the application may specify that they should be consumed in first-in-first-out last-in-first-out order). The user may also supply a tag in a pull request (or they may submit a wildcard tag), and this tag may be used by the data pool to identify the data item that will be consumed. Additional information such as the amount of data requested and consumer's process ID may also be used by the data pool in associating a pull request with a data item. Depending on the model selected by the user, a data pull may consume an entire datum, or it may partially consume a datum allowing subsequent pulls to consume the remainder of the datum.

**[0026]** In an example, when the DPE receives a pull, it performs the following:

a. Extract tag (species identifier), consumer ID, and requested size from the pull request
b. Query the data pool using the pull request parameters.
c. If the data pool finds a matching datum, it provides a buffer pointer and size to the DPE.
d. DPE sends the response to the consumer immediately.
    i. Once a datum has been consumed, the data pool updates the datum by removing all or a portion of the datum from the pool.
    ii. A notification is generated at the consumer when the pull response has been received to notify the consumer that a datum is available. The notification may further contain additional information such as the amount of data received and the species tag.
e. If the data pool cannot find a matching datum, it may try the following:
    i. The DPE sends an empty response to the consumer indicating that no data item matching their request was found.
    ii. The DPE sends a "pending" response to the consumer indicating the DPE has enqueued their request and will respond when a data item becomes available.
    iii. When pull requests are pending at a particular DPE, data pool may attempt to match any new data items against pending requests.

iv. If ordered data are present, it may be necessary to perform this check in the order in which requests were received and before the new data are added to the pool.

v. If ordering is not enabled, attempts to match pending pull requests may be performed in parallel after data have been added to the pool.

f. For pulls marked as pending, a subsequent DPE pull response can take a variety of forms, depending on what is requested by the consumer or supported by the system. For example, the consumer may set up a region of memory allowing the DPE to perform a put operation to transmit the pull response. Alternatively, a push operation can be used to append data to a posted buffer or the data pool at the consumer.

i. Upon completion of the pull response at the consumer, a notification is generated at the consumer.

**[0027]** In some cases, a DPE may be implemented using a "Portals 4" HFI. One-sided pull operations can be supported on a variety of networks through a software implementation of the DPE. Performance enhancements may be achieved through hardware implementations.

**[0028]** On a Portals 4 network, several embodiments of the DPE can be implemented that may be sufficient for a range of application usage models. For example, if only local pushing is supported, the data pool can be implemented as a list of match entries (MEs) that can be posted with one datum per ME. Data can be consumed entirely, or in part if the ME has the PTL_ME_MANAGE_LOCAL option set. A truncation entry is appended to the overflow list to generate a zero byte response if no matching datum is available. Delayed responses are possible through software if a full event is captured at the producer upon truncation. Pull operations may be performed by the consumer through a call to PtlGet(), with the desired species tag embedded into the match bits. The PtlGet() can be sent to the Portal table entry on the matching interface that corresponds to the producer's data pool.

**[0029]** A system and method for providing a fabric data pulling engine will now be described with more particular reference to the attached FIGURES. It should be noted that throughout the FIGURES, certain reference numerals may be repeated to indicate that a particular device or block is wholly or substantially consistent across the FIGURES. This is not, however, intended to imply any particular relationship between the various embodiments disclosed. In certain examples, a genus of elements may be referred to by a particular reference

numeral ("widget 10"), while individual species or examples of the genus may be referred to by a hyphenated numeral ("first specific widget 10-1" and "second specific widget 10-2").

**[0030]** FIGURE 1 is a block diagram of a high-performance computing (HPC) system according to one or more examples of the present specification. HPC system 100 may be any suitable HPC system, including for example a supercomputer, computing cluster, Beowulf cluster, or distributed network, as appropriate to the embodiment.

**[0031]** HPC system 100 includes a plurality of compute nodes. In this case, each node may be any suitable computing device, system, or subsystem. For example, each compute node 110-4 could be a blade server, rack-mount server, or standalone server. Each computer node 110 could be or comprise a system on a chip (SoC), or each compute node 110 could be housed in a standalone chassis.

**[0032]** In the illustrated example, five compute nodes, 110-1, 110-2, 110-3, 110-4, and 110-5 are shown. However, this is merely an illustration of the operational principle. In operation, any number of compute nodes 110 could be provided. For example, HPC system 100 could be a Beowulf cluster built of a small number of commodity, off-the-shelf computers, laptops, or blades. Or HPC system 100 could be a high-end supercomputer, with many thousands of homogeneous nodes providing petaflops of processing power.

**[0033]** Each compute node 110 communicatively couples to a fabric 170 via a host fabric interface (HFI) 260. Specifically, compute node 110-1 couples to fabric 170 via HFI 260-1. Compute node 110-2 communicatively couples to fabric 170 via HFI 260-2. Compute node 110-3 communicatively couples to fabric 170 via HFI 260-3. Compute node 110-4 communicatively couples to fabric 170 via HFI 260-4. Compute node 110-5 communicatively couples to fabric 170 via HFI 260-5.

**[0034]** Compute nodes 110 may be in any suitable configuration, including for example a motherboard-based design in which HFI 260 is an insertable card, such as a PCIe card. Compute node 110 may also be implemented on a multi-chip package, single-board computer, or system-on-a-chip, by way of nonlimiting example.

**[0035]** Fabric 170 may be any suitable fabric for communicatively coupling compute nodes 110 to one another. This may take the form of, for example, Intel® OmniPath™ architecture, Infiniband, Ethernet, or any other suitable fabric.

**[0036]** FIGURE 2 is a block diagram of compute node 110 according to one or more examples of the present specification. Compute node 110 may be any suitable computing device.

In various embodiments, a "computing device" may be or comprise, by way of non-limiting example, a computer, workstation, server, mainframe, virtual machine (whether emulated or on a "bare-metal" hypervisor), embedded computer, embedded controller, embedded sensor, personal digital assistant, laptop computer, cellular telephone, IP telephone, smart phone, tablet computer, convertible tablet computer, computing appliance, network appliance, receiver, wearable computer, handheld calculator, or any other electronic, microelectronic, or microelectromechanical device for processing and communicating data. Any computing device may be designated as a host on the network. Each computing device may refer to itself as a "local host," while any computing device external to it may be designated as a "remote host."

**[0037]** Compute node 110 includes one or more processors 210. In this example, two processors are shown, namely processor 210-1 and processor 210-2. These are communicatively coupled to a local memory 220, having stored therein executable instructions for providing an operating system 222 and at least software portions of a producer/consumer engine 224. Other components of compute node 110 include an optional storage 250, and host fabric interface 260. This architecture is provided by way of example only, and is intended to be non-exclusive and non-limiting. Furthermore, the various parts disclosed are intended to be logical divisions only, and need not necessarily represent physically separate hardware and/or software components. Certain computing devices provide main memory 220 and storage 250, for example, in a single physical memory device, and in other cases, memory 220 and/or storage 250 are functionally distributed across many physical devices. In the case of virtual machines or hypervisors, all or part of a function may be provided in the form of software or firmware running over a virtualization layer to provide the disclosed logical function. In other examples, a device such as a host fabric interface 260 may provide only the minimum hardware interfaces necessary to perform its logical operation, and may rely on a software driver to provide additional necessary logic. Thus, each logical block disclosed herein is broadly intended to include one or more logic elements configured and operable for providing the disclosed logical operation of that block. As used throughout this specification, "logic elements" may include hardware, external hardware (digital, analog, or mixed-signal), software, reciprocating software, services, drivers, interfaces, components, modules, algorithms, sensors, components, firmware, microcode, programmable logic, or objects that can coordinate to achieve a logical operation.

**[0038]** In an example, processors 210 are communicatively coupled to memory 220 via memory bus 270-3, which may be for example a direct memory access (DMA) bus by way of

example, though other memory architectures are possible, including ones in which memory 220 communicates with processor 210 via system bus 270-1 or some other bus. Processor 210 may be communicatively coupled to other devices via a system bus 270-1. As used throughout this specification, a "bus" includes any wired or wireless interconnection line, network, connection, bundle, single bus, multiple buses, crossbar network, single-stage network, multistage network or other conduction medium operable to carry data, signals, or power between parts of a computing device, or between computing devices. It should be noted that these uses are disclosed by way of non-limiting example only, and that some embodiments may omit one or more of the foregoing buses, while others may employ additional or different buses.

**[0039]** In various examples, a "processor" may include any combination of logic elements operable to execute instructions, whether loaded from memory, or implemented directly in hardware, including by way of non-limiting example a microprocessor, digital signal processor, field-programmable gate array, graphics processing unit, programmable logic array, application-specific integrated circuit, or virtual machine processor. In certain architectures, a multi-core processor may be provided, in which case processor 210 may be treated as only one core of a multi-core processor, or may be treated as the entire multi-core processor, as appropriate. In some embodiments, one or more co-processor may also be provided for specialized or support functions.

**[0040]** Processor 210 may be connected to memory 220 in a DMA configuration via DMA bus 270-3. To simplify this disclosure, memory 220 is disclosed as a single logical block, but in a physical embodiment may include one or more blocks of any suitable volatile or non-volatile memory technology or technologies, including for example DDR RAM, SRAM, DRAM, cache, L1 or L2 memory, on-chip memory, registers, flash, ROM, optical media, virtual memory regions, magnetic or tape memory, or similar. In certain embodiments, memory 220 may comprise a relatively low-latency volatile main memory, while storage 250 may comprise a relatively higher-latency non-volatile memory. However, memory 220 and storage 250 need not be physically separate devices, and in some examples may represent simply a logical separation of function. It should also be noted that although DMA is disclosed by way of non-limiting example, DMA is not the only protocol consistent with this specification, and that other memory architectures are available.

**[0041]** Storage 250 may be any species of memory 220, or may be a separate device. Storage 250 may include one or more non-transitory computer-readable mediums, including by

way of non-limiting example, a hard drive, solid-state drive, external storage, redundant array of independent disks (RAID), network-attached storage, optical storage, tape drive, backup system, cloud storage, or any combination of the foregoing. Storage 250 may be, or may include therein, a database or databases or data stored in other configurations, and may include a stored copy of operational software such as operating system 222 and software portions of producer/consumer engine 224. Many other configurations are also possible, and are intended to be encompassed within the broad scope of this specification.

**[0042]** Host fabric interface 260 may be provided to communicatively couple compute node 110 to a wired or wireless network. A "network," as used throughout this specification, may include any communicative platform operable to exchange data or information within or between computing devices, including by way of non-limiting example, an ad-hoc local network, an internet architecture providing computing devices with the ability to electronically interact, a plain old telephone system (POTS), which computing devices could use to perform transactions in which they may be assisted by human operators or in which they may manually key data into a telephone or other suitable electronic equipment, any packet data network (PDN) offering a communications interface or exchange between any two nodes in a system, or any local area network (LAN), metropolitan area network (MAN), wide area network (WAN), wireless local area network (WLAN), virtual private network (VPN), intranet, or any other appropriate architecture or system that facilitates communications in a network or telephonic environment.

**[0043]** Producer/consumer engine 224, in one example, is operable to carry out computer-implemented methods as described in this specification, including acting as either a producer of data or a consumer of data. These relationships are described in more detail in connection with FIGURE 3.

**[0044]** Producer/consumer engine 224 may include one or more tangible non-transitory computer-readable mediums having stored thereon executable instructions operable to instruct a processor to provide a producer/consumer engine 224. As used throughout this specification, an "engine" includes any combination of one or more logic elements, of similar or dissimilar species, operable for and configured to perform one or more methods provided by the engine. Thus, producer/consumer engine 224 may comprise one or more logic elements configured to provide methods as disclosed in this specification. In some cases, producer/consumer engine 224 may include a special integrated circuit designed to carry out a method or a part thereof, and may also include software instructions operable to instruct a processor to perform the

method. In some cases, producer/consumer engine 224 may run as a "daemon" process. A "daemon" may include any program or series of executable instructions, whether implemented in hardware, software, firmware, or any combination thereof that runs as a background process, a terminate-and-stay-resident program, a service, system extension, control panel, bootup procedure, BIOS subroutine, or any similar program that operates without direct user interaction. In certain embodiments, daemon processes may run with elevated privileges in a "driver space," or in ring 0, 1, or 2 in a protection ring architecture. It should also be noted that producer/consumer engine 224 may also include other hardware and software, including configuration files, registry entries, and interactive or user-mode software by way of non-limiting example.

**[0045]** In one example, producer/consumer engine 224 includes executable instructions stored on a non-transitory medium operable to perform a method according to this specification. At an appropriate time, such as upon booting compute node 110 or upon a command from operating system 222 or a user 120, processor 210 may retrieve a copy of the instructions from storage 250 and load it into memory 220. Processor 210 may then iteratively execute the instructions of producer/consumer engine 224 to provide the desired method.

**[0046]** Depending on the role that compute node 110 plays in HPC system 100, producer/consumer engine 224 may operate as a data producer engine or as a data consumer engine. As a data producer engine, producer/consumer engine 224 may be configured to operate on a data set and produce data for consumption by other nodes, or a data producer engine may be configured to aggregate outputs from other processes and compile them into suitable inputs for different processes.

**[0047]** As a data consumer engine, producer/consumer engine 224 may be configured to consume inputs from other processes. To provide just one example, a common task for parallel computing is calculating discrete values in a sequence of parallel difference equation, where each stage operates on an output of a previous stage. Thus, at each stage, a data producer engine may collect outputs from a previous stage, place them into a data pool, and then receive data requests from a plurality of data consumer engines, each running an identical difference equation. In that case, the data producer engine provides the data to the data consumer engines, and then again collects the outputs.

**[0048]** FIGURE 3 is a block diagram of HPC system 100 illustrating that certain nodes can act as producers or consumers as discussed above. In this example, node 340 acts as a data

producer. Node 350 acts as a data consumer. In this example, producer 340 and consumer 350 are illustrated by way of example as being on two separate modes. It should be understood, however, that producer 340 and consumer 350 could be on the same compute node 110.

**[0049]** HPC system 100 may be configured to run a massively parallel computing task. This may be a task that lends itself to being divided into smaller memory chunks, each of which can be handled by a separate compute node 110. In such a case, producer 340 may prepare data packets for parceling out to compute nodes 110.

**[0050]** Consumer 350 may have a portion of code that is can use to operate on data. That same code may be reproduced across many consumers 350, even up to tens or hundreds of thousands of consumers 350. Thus, as illustrated herein, complete two-way synchronization between one or more producers 340 and one or more consumers 350 may be impractical. On the other hand, one-way communication also carries certain challenges as discussed above. Thus the present specification provides a fabric integrated data pulling engine that may be considered "one-and-a-half" way communication.

**[0051]** Specifically, consumer 350 need not be aware of or care which section of data it is operating on. Rather, consumer 350 runs a procedure that has a set of given inputs, and that produces a set of outputs from those inputs. Those outputs may be the input to a next stage. Thus, consumer 350 need only concern itself with getting a datum to operate on, without worrying about how that datum fits into the overall picture of the computation. The data structure, meaning the inputs and the outputs that consumer 350 needs to operate on, is referred to herein as a species of data. In other words, there is a particular datum, which should be understood in this context to mean a structured packet, which may include a particular format and types of values, that consumer 350 needs to receive. When consumer 350 requests a datum, it may request a specific species, or it may wildcard at least some entries. Producer 340 may produce one or many data that fit the profile of the particular species that consumer 350 needs.

**[0052]** Thus, according to one example, consumer 350 sends a data request to producer 340. Producer 340 may have a pool of data of the appropriate species for consumer 350. Thus, producer 340 may select a datum from those data and provide that datum to consumer 350 according to the methods disclosed in this specification. Consumer 350 can then operate on that datum, complete its function, and then perform any reporting function that it needs to do. Once consumer 350 has completed its task, it may be prepared to receive a new datum. Thus,

consumer 350 may again send a request to producer 340. Producer 340 may then provide a new datum to consumer 350.

**[0053]** If data are not available when consumer 350 requests a datum, then producer 340 may queue the request. Requests can optionally be queued in first-in-first-out order, in which case queued requests may be handled before new data are added to the pool. In another embodiment, queued requests are unordered, and are handled in parallel once sufficient data have been added to the pool. In other embodiments, any other suitable queuing method may be used, such as last-in-first-out and round robin by way of nonlimiting example. In some embodiments, producer 340 sends a signal to consumer 350 indicating that the request has been received, and that it has been queued but not yet handled.

**[0054]** It should be noted that when producer 340 initially produces the data, producer 340 does not need to know which specific compute node 110 that the data will go to. Rather, producer 340 can simply designate the species of each new datum, store it in memory, and notify DPE 420 (FIGURE 4). Once the datum is stored in memory, data pulling engine 420 of HFI 260 can publish a notice that data of the species are available. The data pulling engine can then handle any additional messaging concerning those data.

**[0055]** FIGURE 4 is a block diagram of an HFI 260 according to one or more examples of the present specification. In this case, HFI 260 includes a bus interface 470, a fabric interface 480, and a DMA interface 420. Fabric interface 480 is to communicatively couple HFI 260 to fabric 170. Bus interface 470 is to communicatively couple HFI 260 to a local bus of the compute node, such as bus 270 of FIGURE 2. And DMA 420 is to communicatively couple HFI 260 to a DMA such as DMA 270-3 of FIGURE 2.

**[0056]** HFI 260 also includes a network interface engine 410. Network interface engine 410 provides traditional input/output operations and management of the fabric interface. Thus in this respect, network interface engine 410 may be similar or identical to the function performed by an HFI in certain existing architectures.

**[0057]** HFI 260 also includes data pulling engine 420. Data pulling engine 420 provides methods according to the present specification. Data pulling engine 420 may be any suitable engine, including any combination of hardware, software, and/or firmware as discussed above.

**[0058]** HFI 260 may also include a data pool 460. Data pool 460 may include data that can be provided to consumers according to the methods of the present specification. As illustrated in FIGURE 5, data pool 460 may include data of a plurality of different species. In the

example of FIGURE 5, data pool 460 includes data species 1 510, and data species 2 520. Thus, when a consumer 350 requests data, it may specify the species of data that it is interested in. If a consumer 350 requests data of species 1 510, then data pulling engine 420 may select any datum from data species 510 (for example, according to an algorithm such as FIFO or LIFO), and provide that datum to consumer 350. If a consumer 350 requests a datum of species 2, then data pulling engine 420 may select any suitable datum of species 2 520, and provide that datum to consumer 350. It should be noted that data pool 460 may be the actual data, or it may simply be notifications that instruct data pulling engine 420 where to retrieve data from memory 220.

**[0059]** FIGURE 6 is a flowchart of a method 600 performed by a producer engine 224 according to one or more examples of the present specification. In this example, at block 610, producer engine 224 generates data.

**[0060]** In block 620, producer engine 224 stores the produced data in local memory 220.

**[0061]** In block 630, producer engine 224 publishes the data to data pulling engine 420 of HFI 260. In an example, publishing to data pulling engine 420 comprises notifying data pulling engine 420 that data are available for sending to a requesting consumer 350. When producer engine 224 publishes the data, data pulling engine 420 may either access it from its original location, or may copy it to a local buffer.

**[0062]** Publishing does not necessarily imply broadcasting data availability to consumers 350, as may be done in a true "two-way" exchange. Rather, producer engine 224 merely notifies data pulling engine 420 that data are available, and is then essentially done with its part of the transaction. Thus, control can pass back to block 610, where producer 340 continues producing data without regard to what data pulling engine 420 does with those data. In the meantime, data pulling engine 420 holds onto the data in its data pool until it receives a data request from a consumer 350.

**[0063]** In some embodiments, it is possible that data pulling engine 420 could notify producer 340 that its data pool is full, as may be the case if consumers 350 are not ready to retrieve data. In that case, producer 340 may suspend production of data, and either periodically poll data pulling engine 420, or wait for a signal from data pulling engine 420 that it is ready to receive more data.

**[0064]** FIGURE 7 is a flowchart of a method 700 performed by data pulling engine 420 of HFI 260 according to one or more examples of the present specification.

**[0065]** At block 710, data pulling engine 420 of HFI 260 receives from producer engine 224 published data (if any). These may be data that come from data pool 460.

**[0066]** In decision block 720, if data pulling engine 420 does not receive a pull request, then it continues to receive published data as before.

**[0067]** If on the other hand, data pulling engine 420 receives a pull request 722, then control may proceed to block 730. It should be noted that pull request may include a node identifier for a consumer 350 that is requesting the data. Pull requests 722 may also include a species identifier for the species of data that is being requested. Pull requests 722 may also include any other header information or other packet format that is suitable for their purpose.

**[0068]** In decision block 740, data pulling engine 420 checks whether there are data available in the data pool of the type requested.

**[0069]** In block 750, if no data are available, then in an embodiment, data pulling engine 420 may queue pull request 722. Optionally, data pulling engine 420 may send a signal to consumer 350 indicating that the request cannot be serviced immediately, but that the request has been queued. If ordering is enabled, then queued requests may be handled in a first-in-first-out order. In that case, queued requests may be handled before new data are added to the pool. If ordering is not enabled, then attempts to match pending pull requests may be performed in parallel after data have been added to the pool.

**[0070]** In block 760, data pulling engine 420 forwards the requested data to the requesting consumer node 350. To do this, data pulling engine 420 specifically directs the message to the node ID of the requesting consumer 350. It should also be noted again, that datum it sends may simply be a datum selected from data pool 460, and need not be a specific datum selected or requested by consumer 350.

**[0071]** In block 770, data pulling engine 420 unpublishes the data that it sent in block 760. This indicates that the datum is no longer available for sending to a consumer 350.

**[0072]** In block 780, data pulling engine 420 releases the memory location of the sent datum. Releasing the memory location may include simply marking the memory location as available for use again. In some cases where security is of a premium concern, this may also include zeroing the memory, or otherwise overwriting or erasing it.

**[0073]** In block 799, the method is done.

**[0074]** FIGURE 8 is a flowchart of a method 800 performed by a consumer engine 224 according to one or more examples of the present specification. When a compute node 110 is

acting as a consumer 350, it may send a message to a producer requesting a species of data. This message may include a node ID for consumer 350. Note that this message need not include a node ID for producer 340. Rather, the availability of data may be published or advertised by data pulling engine 420, and consumer 350 may simply respond by requesting a datum for itself.

**[0075]** In block 820, consumer engine 224 receives the requested datum.

**[0076]** In block 830, consumer engine 224 consumes the datum.

**[0077]** In block 899, the method is done. It should be noted that in certain embodiments, the method 800 a FIGURE 8 may be enabled by a network interface engine 410 that need not be specially modified as in the case of data pulling engine 420.

**[0078]** The foregoing outlines features of several embodiments so that those skilled in the art may better understand various aspects of the present disclosure. Those skilled in the art should appreciate that they may readily use the present disclosure as a basis for designing or modifying other processes and structures for carrying out the same purposes and/or achieving the same advantages of the embodiments introduced herein. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the present disclosure, and that they may make various changes, substitutions, and alterations herein without departing from the spirit and scope of the present disclosure.

**[0079]** All or part of any hardware element disclosed herein may readily be provided in a system-on-a-chip (SoC), including central processing unit (CPU) package. An SoC represents an integrated circuit (IC) that integrates components of a computer or other electronic system into a single chip. Thus, for example, client devices 110 or server devices 300 may be provided, in whole or in part, in an SoC. The SoC may contain digital, analog, mixed-signal, and radio frequency functions, all of which may be provided on a single chip substrate. Other embodiments may include a multi-chip-module (MCM), with a plurality of chips located within a single electronic package and configured to interact closely with each other through the electronic package. In various other embodiments, the computing functionalities disclosed herein may be implemented in one or more silicon cores in Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), and other semiconductor chips.

**[0080]** Note also that in certain embodiment, some of the components may be omitted or consolidated. In a general sense, the arrangements depicted in the figures may be more logical in their representations, whereas a physical architecture may include various permutations, combinations, and/or hybrids of these elements. It is imperative to note that countless possible

design configurations can be used to achieve the operational objectives outlined herein. Accordingly, the associated infrastructure has a myriad of substitute arrangements, design choices, device possibilities, hardware configurations, software implementations, and equipment options.

**[0081]** In a general sense, any suitably-configured processor, such as processor 210, can execute any type of instructions associated with the data to achieve the operations detailed herein. Any processor disclosed herein could transform an element or an article (for example, data) from one state or thing to another state or thing. In another example, some activities outlined herein may be implemented with fixed logic or programmable logic (for example, software and/or computer instructions executed by a processor) and the elements identified herein could be some type of a programmable processor, programmable digital logic (for example, a field programmable gate array (FPGA), an erasable programmable read only memory (EPROM), an electrically erasable programmable read only memory (EEPROM)), an ASIC that includes digital logic, software, code, electronic instructions, flash memory, optical disks, CD-ROMs, DVD ROMs, magnetic or optical cards, other types of machine-readable mediums suitable for storing electronic instructions, or any suitable combination thereof.

**[0082]** In operation, a storage such as storage 250 may store information in any suitable type of tangible, non-transitory storage medium (for example, random access memory (RAM), read only memory (ROM), field programmable gate array (FPGA), erasable programmable read only memory (EPROM), electrically erasable programmable ROM (EEPROM), etc.), software, hardware (for example, processor instructions or microcode), or in any other suitable component, device, element, or object where appropriate and based on particular needs. Furthermore, the information being tracked, sent, received, or stored in a processor could be provided in any database, register, table, cache, queue, control list, or storage structure, based on particular needs and implementations, all of which could be referenced in any suitable timeframe. Any of the memory or storage elements disclosed herein, such as memory 220 and storage 250, should be construed as being encompassed within the broad terms 'memory' and 'storage,' as appropriate. A non-transitory storage medium herein is expressly intended to include any non-transitory special-purpose or programmable hardware configured to provide the disclosed operations, or to cause a processor such as processor 210 to perform the disclosed operations.

**[0083]** Computer program logic implementing all or part of the functionality described herein is embodied in various forms, including, but in no way limited to, a source code form, a computer executable form, machine instructions or microcode, programmable hardware, and various intermediate forms (for example, forms generated by an assembler, compiler, linker, or locator). In an example, source code includes a series of computer program instructions implemented in various programming languages, such as an object code, an assembly language, or a high-level language such as OpenCL, FORTRAN, C, C++, JAVA, or HTML for use with various operating systems or operating environments, or in hardware description languages such as Spice, Verilog, and VHDL. The source code may define and use various data structures and communication messages. The source code may be in a computer executable form (e.g., via an interpreter), or the source code may be converted (e.g., via a translator, assembler, or compiler) into a computer executable form, or converted to an intermediate form such as byte code. Where appropriate, any of the foregoing may be used to build or describe appropriate discrete or integrated circuits, whether sequential, combinatorial, state machines, or otherwise.

**[0084]** In one example embodiment, any number of electrical circuits of the FIGURES may be implemented on a board of an associated electronic device. The board can be a general circuit board that can hold various components of the internal electronic system of the electronic device and, further, provide connectors for other peripherals. More specifically, the board can provide the electrical connections by which the other components of the system can communicate electrically. Any suitable processor and memory can be suitably coupled to the board based on particular configuration needs, processing demands, and computing designs. Other components such as external storage, additional sensors, controllers for audio/video display, and peripheral devices may be attached to the board as plug-in cards, via cables, or integrated into the board itself. In another example, the electrical circuits of the FIGURES may be implemented as stand-alone modules (e.g., a device with associated components and circuitry configured to perform a specific application or function) or implemented as plug-in modules into application specific hardware of electronic devices.

**[0085]** Note that with the numerous examples provided herein, interaction may be described in terms of two, three, four, or more electrical components. However, this has been done for purposes of clarity and example only. It should be appreciated that the system can be consolidated or reconfigured in any suitable manner. Along similar design alternatives, any of the illustrated components, modules, and elements of the FIGURES may be combined in various

possible configurations, all of which are within the broad scope of this specification. In certain cases, it may be easier to describe one or more of the functionalities of a given set of flows by only referencing a limited number of electrical elements. It should be appreciated that the electrical circuits of the FIGURES and its teachings are readily scalable and can accommodate a large number of components, as well as more complicated/sophisticated arrangements and configurations. Accordingly, the examples provided should not limit the scope or inhibit the broad teachings of the electrical circuits as potentially applied to a myriad of other architectures.

**[0086]** Numerous other changes, substitutions, variations, alterations, and modifications may be ascertained to one skilled in the art and it is intended that the present disclosure encompass all such changes, substitutions, variations, alterations, and modifications as falling within the scope of the appended claims. In order to assist the United States Patent and Trademark Office (USPTO) and, additionally, any readers of any patent issued on this application in interpreting the claims appended hereto, Applicant wishes to note that the Applicant: (a) does not intend any of the appended claims to invoke paragraph six (6) of 35 U.S.C. section 112 (pre-AIA) or paragraph (f) of the same section (post-AIA), as it exists on the date of the filing hereof unless the words "means for" or "steps for" are specifically used in the particular claims; and (b) does not intend, by any statement in the specification, to limit this disclosure in any way that is not otherwise expressly reflected in the appended claims.

Example Implementations

**[0087]** There is disclosed in example 1, a compute node, comprising: first one or more logic elements for providing a data producer engine to produce a datum; and a host fabric interface to communicatively couple the compute node to a fabric, the fabric interface comprising second one or more logic elements for providing a data pulling engine, the data pulling engine to: publish the datum as available; receive a pull request for the datum, the pull request comprising a node identifier for a data consumer; and send the datum to the data consumer via the fabric.

**[0088]** There is disclosed in example 2, the compute node of example 1, wherein the compute node is to store the datum in a local memory at a memory location.

**[0089]** There is disclosed in example 3, the compute node of example 2, wherein the data pulling engine is further to release the memory location after sending the data.

**[0090]** There is disclosed in example 4, the compute node of example 3, wherein releasing the memory location comprises marking the memory location as available for reuse.

**[0091]** There is disclosed in example 5, the compute node of example 2, wherein the datum is of a first species, and the local memory is further to receive a second datum of a second species.

**[0092]** There is disclosed in example 6, the compute node of example 5, wherein the pull request comprises a species designation, and wherein data pulling engine is to select between the datum of the first species and the second datum of the second species based at least in part on the species designation.

**[0093]** There is disclosed in example 7, the compute node of example 2, wherein the datum is of a first species, the local memory is to receive a plurality of data of the first species, and wherein sending the datum to the data consumer comprises sending any one of the data of the first species to the data consumer.

**[0094]** There is disclosed in example 8, the compute node of any of examples 1 – 7, wherein the data pulling engine is further to unpublish the data after sending the data.

**[0095]** There is disclosed in example 9, the compute node of any of examples 1 – 7, wherein the data pulling engine is further to: send a pull request to a second data pulling engine; and receive a datum from the second data pulling engine.

**[0096]** There is disclosed in example 10, a host fabric interface, comprising: first one or more logic elements for providing a network interface engine to communicatively couple the host fabric interface to a fabric; second one or more logic elements for providing a bus interface engine to communicatively couple the host fabric interface to a data producer engine; and third one or more logic elements comprising a data pulling engine, the data pulling engine to: determine that a datum is available from the data producer engine and is stored in a memory location of a local memory; publish the datum as available via the fabric; receive a pull request for the datum via the fabric, the pull request comprising a node identifier for a data consumer; and send the datum to the data consumer via the fabric.

**[0097]** There is disclosed in example 11, the host fabric interface of example 10, wherein the data producer is to store the datum in a local memory at a memory location.

**[0098]** There is disclosed in example 12, the host fabric interface of example 11, wherein the data pulling engine is further to release the memory location after sending the data.

**[0099]** There is disclosed in example 13, the host fabric interface of example 12, wherein releasing the memory location comprises marking the memory location as available for reuse.

**[0100]** There is disclosed in example 14, the host fabric interface of example 11, wherein the datum is of a first species, and the local memory is further to receive a second datum of a second species.

**[0101]** There is disclosed in example 15, the host fabric interface of example 14, wherein the pull request comprises a species designation, and wherein data pulling engine is to select between the datum of the first species and the second datum of the second species based at least in part on the species designation.

**[0102]** There is disclosed in example 16, the host fabric interface of example 11, wherein the datum is of a first species, the local memory is to receive a plurality of data of the first species, and wherein sending the datum to the data consumer comprises sending any one of the data of the first species to the data consumer.

**[0103]** There is disclosed in example 17, the host fabric interface of any of examples 10 – 16, wherein the data pulling engine is further to unpublish the data after sending the data.

**[0104]** There is disclosed in example 18, the host fabric interface of any of examples 10 – 16, wherein the data pulling engine is further to: send a pull request to a second data pulling engine; and receive a datum from the second data pulling engine.

**[0105]** There is disclosed in example 19, a method of providing a data pulling engine, comprising: communicatively coupling to a fabric; communicatively coupling to a data producer; and determining that a datum is available from the data producer and is stored in a memory location of a local memory; publishing the datum as available via the fabric; receive a pull request for the datum via the fabric, the pull request comprising a node identifier for a data consumer; and send the datum to the data consumer via the fabric.

**[0106]** There is disclosed in example 20, the method of example 19, further comprising releasing the memory location after sending the data.

**[0107]** There is disclosed in example 21, the method of example 20, wherein releasing the memory location comprises marking the memory location as available for reuse.

**[0108]** There is disclosed in example 22, the method of example 21, wherein the datum is of a first species, and the local memory is further to receive a second datum of a second species.

**[0109]** There is disclosed in example 23, the method of any of examples 19 – 22, wherein the pull request comprises a species designation, and wherein data pulling engine is to select

between the datum of the first species and the second datum of the second species based at least in part on the species designation.

**[0110]** There is disclosed in example 24, the method of any of examples 19 – 22, wherein the datum is of a first species, the local memory is to receive a plurality of data of the first species, and wherein sending the datum to the data consumer comprises sending any one of the data of the first species to the data consumer.

**[0111]** There is disclosed in example 25, the method of any of examples 19 – 22, further comprising unpublishing the data after sending the data.

**[0112]** There is disclosed in example 26, an apparatus comprising means for performing the method of any of examples 19 – 25.

**[0113]** There is disclosed in example 27, the apparatus of example 26, wherein the means for performing the method comprise a processor and a memory.

**[0114]** There is disclosed in example 28, the apparatus of Example 27, wherein the memory comprises machine-readable instructions, that when executed cause the apparatus to perform the method of any of examples 19 – 25.

**[0115]** There is disclosed in example 29, the apparatus of any of Examples 26 – 28, wherein the apparatus is a computing system.

**[0116]** There is disclosed in example 30, at least one computer readable medium comprising instructions that, when executed, implement a method or realize an apparatus as described in any of examples 19 – 29.

**[0117]** There is disclosed in example 31, a method of providing a data pulling engine, comprising: communicatively coupling to a fabric; communicatively coupling to a data producer; and determining that a datum is available from the data producer and is stored in a memory location of a local memory, wherein the datum is of a first species, and wherein the memory is further to receive a datum of a second species; publishing the datum as available via the fabric; receiving a pull request for a datum of the first species via the fabric, the pull request comprising a node identifier for a data consumer; and sending the datum to the data consumer via the fabric.

**[0118]** There is disclosed in example 32, the method of example 31, further comprising: generating a datum of the first species; storing the datum in the memory; and publishing the datum to a data pulling engine.

**[0119]** There is disclosed in example 33, the method of example 31, further comprising: sending a message to a producer requesting a datum of the first species, the message comprising a node identifier; receiving the requested datum; and consuming the datum.

**[0120]** There is disclosed in example 34, an apparatus comprising means for performing the method of any of examples 31 – 33.

**[0121]** There is disclosed in example 35, the apparatus of example 34, wherein the means for performing the method comprise a processor and a memory.

**[0122]** There is disclosed in example 36, the apparatus of Example 35, wherein the memory comprises machine-readable instructions, that when executed cause the apparatus to perform the method of any of examples 31 – 33.

**[0123]** There is disclosed in example 37, the apparatus of any of Examples 34 – 36, wherein the apparatus is a computing system.

**[0124]** There is disclosed in example 38, at least one computer readable medium comprising instructions that, when executed, implement a method or realize an apparatus as described in any of examples 31 – 37.

**[0125]** There is also disclosed in an example, a multichip package comprising the compute node or host fabric interface of any preceding example, wherein the producer engine is provided on a first chip and the data pulling engine is provided on a second chip.

**[0126]** There is also disclosed in an example, a system-on-a-chip comprising the compute node of example 1, wherein the producer engine is provided on a first subassembly and the data pulling engine is provided on a second subassembly.

**[0127]** There is also disclosed in an example, an apparatus comprising the compute node of example 1, wherein the host fabric interface comprises a pluggable network card.

**[0128]** There is also disclosed in an example, a high-performance computing system comprising a plurality of the compute node of example 1.

Claims:

1. A compute node, comprising:

one or more first logic elements for providing a data producer engine to produce a datum, and to publish the datum as available; and

a fabric interface to communicatively couple the compute node to a fabric, the host fabric interface comprising one or more second logic elements for providing a data pulling engine, the data pulling engine to:

receive a pull request for the datum, the pull request comprising a node identifier for a data consumer; and

send the datum to the data consumer via the fabric.

2. The compute node of claim 1, wherein the compute node is to store the datum in a local memory at a memory location.

3. The compute node of claim 2, wherein the data pulling engine is further to release the memory location after sending the data.

4. The compute node of claim 3, wherein releasing the memory location comprises marking the memory location as available for reuse.

5. The compute node of claim 2, wherein the datum is of a first species, and the local memory is further to receive a second datum of a second species.

6. The compute node of claim 5, wherein the pull request comprises a species designation, and wherein data pulling engine is to select between the datum of the first species and the second datum of the second species based at least in part on the species designation.

7. The compute node of claim 2, wherein the datum is of a first species, the local memory is to receive a plurality of data of the first species, and wherein sending the datum to the data consumer comprises sending any one of the data of the first species to the data consumer.

8. The compute node of any of the claims of 1 – 7, wherein the data pulling engine is further to unpublish the data after sending the data.

9. The compute node of any of the claims of 1 – 7, wherein the data pulling engine is further to:

send a pull request to a second data pulling engine; and

receive a datum from the second data pulling engine.

10. A multichip package comprising the compute node of any of the claims of 1 – 7, wherein the producer engine is provided on a first chip and the data pulling engine is provided on a second chip.

11. A system-on-a-chip comprising the compute node of any of the claims of 1 – 7, wherein the producer engine is provided on a first subassembly and the data pulling engine is provided on a second subassembly.

12. An apparatus comprising the compute node of any of the claims of 1 – 7, wherein the host fabric interface comprises a pluggable network card.

13. A high-performance computing system comprising a plurality of the compute node of any of the claims of 1 – 7.

14. A host fabric interface, comprising:

one or more first logic elements for providing a network interface engine to communicatively couple the host fabric interface to a fabric;

one or more second logic elements for providing a bus interface engine to communicatively couple the fabric interface to a data producer engine; and

one or more third logic elements for providing a data pulling engine, the data pulling engine to:

determine that a datum is available from the data producer engine and is stored in a memory location of a local memory;

publish the datum as available via the fabric;

receive a pull request for the datum via the fabric, the pull request comprising a node identifier for a data consumer; and

send the datum to the data consumer via the fabric.

15. The host fabric interface of claim 14, wherein the data producer is to store the datum in a local memory at a memory location.

16. The host fabric interface of claim 15, wherein the data pulling engine is further to release the memory location after sending the data.

17. The host fabric interface of claim 16, wherein releasing the memory location comprises marking the memory location as available for reuse.

18. The host fabric interface of claim 15, wherein the datum is of a first species, and the local memory is further to receive a second datum of a second species.

19. The host fabric interface of claim 18, wherein the pull request comprises a species designation, and wherein data pulling engine is to select between the datum of the first species and the second datum of the second species based at least in part on the species designation.

20. The host fabric interface of claim 15, wherein the datum is of a first species, the local memory is to receive a plurality of data of the first species, and wherein sending the datum to the data consumer comprises sending any one of the data of the first species to the data consumer.

21. The host fabric interface of any of claims 14 – 20, wherein the data pulling engine is further to unpublish the data after sending the data.

22. The host fabric interface of any of claims 14 – 20, wherein the data pulling engine is further to:

send a pull request to a second data pulling engine; and

receive a datum from the second data pulling engine.

23. A method of providing data, comprising:

determining that a datum is available from a data producer and is stored in a memory location of a local memory;

publishing the datum as available via a fabric;

receive a pull request for the datum via the fabric, the pull request comprising a node identifier for a data consumer; and

send the datum to the data consumer via the fabric.

24. The method of claim 23, further comprising releasing the memory location after sending the data.

25. The method of claim 24, wherein releasing the memory location comprises marking the memory location as available for reuse.

26. The method of claim 25, wherein the datum is of a first species, and the local memory is further to receive a second datum of a second species.

27. The method of any of claims 23 – 26, wherein the pull request comprises a species designation, and wherein data pulling engine is to select between the datum of the first species and the second datum of the second species based at least in part on the species designation.

28. The method of any of claims 23 – 26, wherein the datum is of a first species, the local memory is to receive a plurality of data of the first species, and wherein sending the datum to the data consumer comprises sending any one of the data of the first species to the data consumer.

29. The method of any of claims 23 – 26, further comprising unpublishing the data after sending the data.

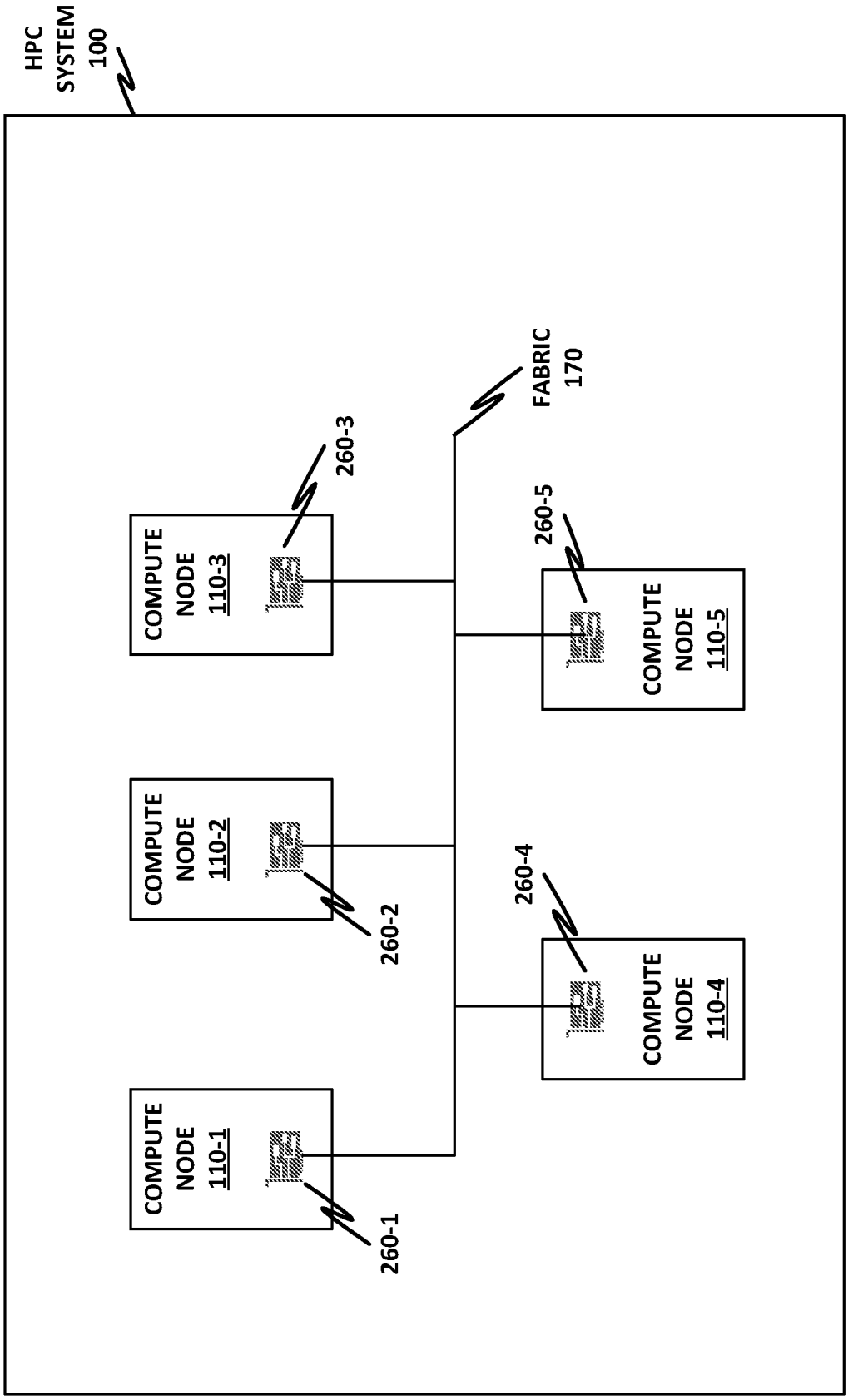
HPC SYSTEM 100
FABRIC 170
COMPUTE NODE 110-1
COMPUTE NODE 110-2
COMPUTE NODE 110-3
COMPUTE NODE 110-4
COMPUTE NODE 110-5
260-1
260-2
260-3
260-4
260-5

Fig. 1

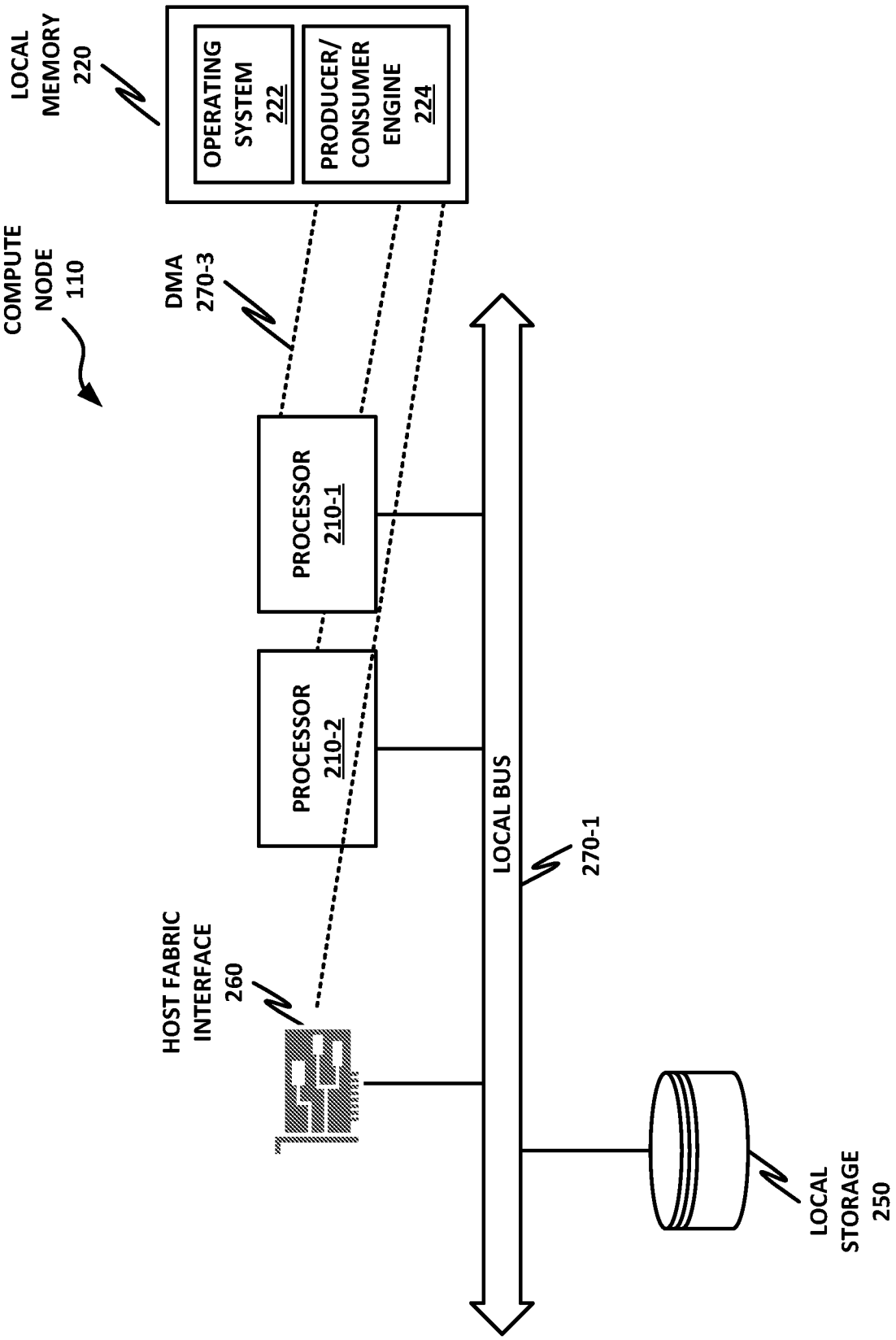
COMPUTE NODE 110
LOCAL MEMORY 220
OPERATING SYSTEM 222
PRODUCER/ CONSUMER ENGINE 224
DMA 270-3
PROCESSOR 210-1
PROCESSOR 210-2
LOCAL BUS 270-1
HOST FABRIC INTERFACE 260
LOCAL STORAGE 250

*Fig. 2*

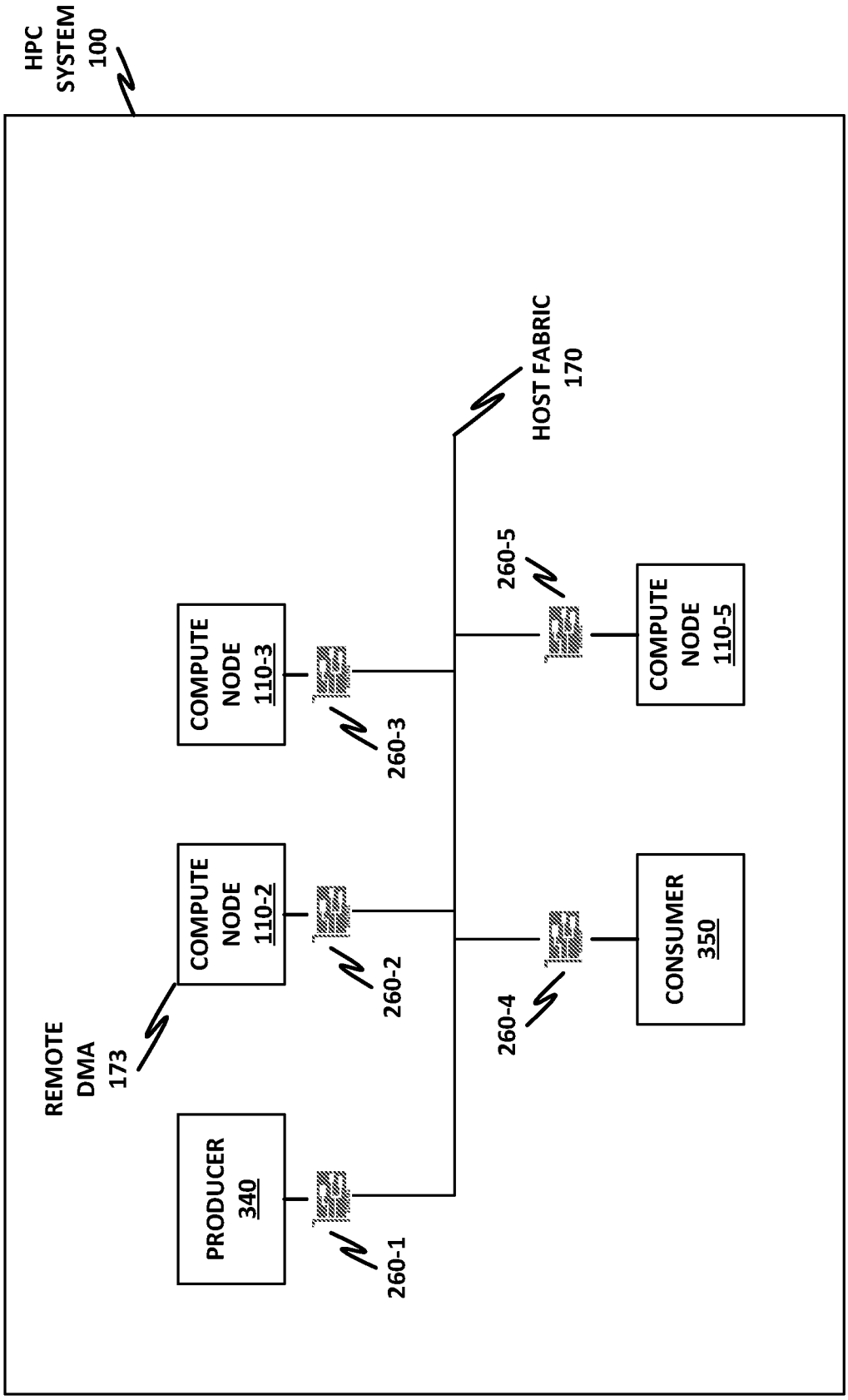
HPC SYSTEM 100
HOST FABRIC 170
REMOTE DMA 173
PRODUCER 340
COMPUTE NODE 110-2
COMPUTE NODE 110-3
CONSUMER 350
COMPUTE NODE 110-5
260-1
260-2
260-3
260-4
260-5

*Fig. 3*

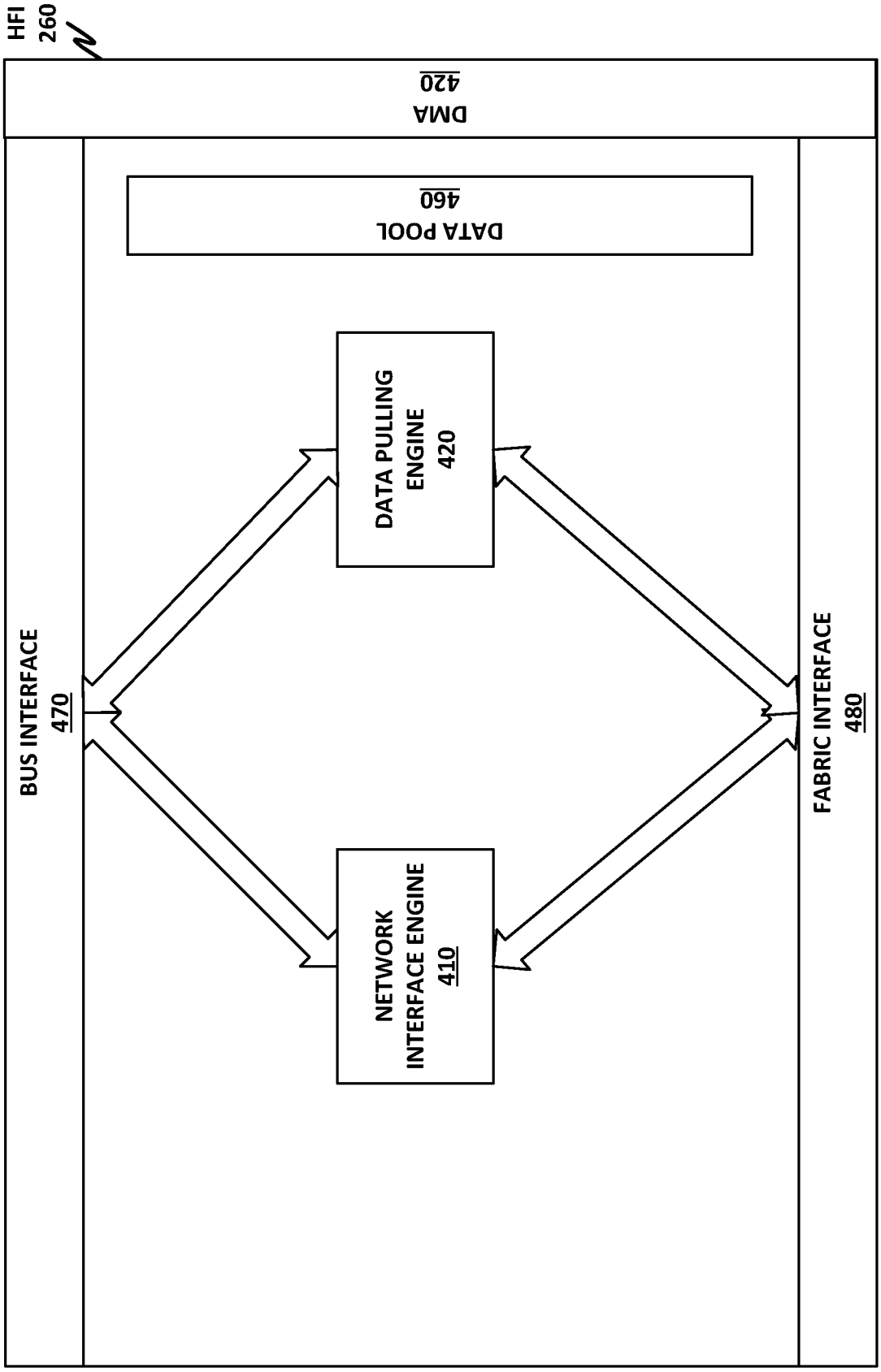
HFI
260
DMA
420
DATA POOL
460
DATA PULLING
ENGINE
420
BUS INTERFACE
470
FABRIC INTERFACE
480
NETWORK
INTERFACE ENGINE
410

*Fig. 4*

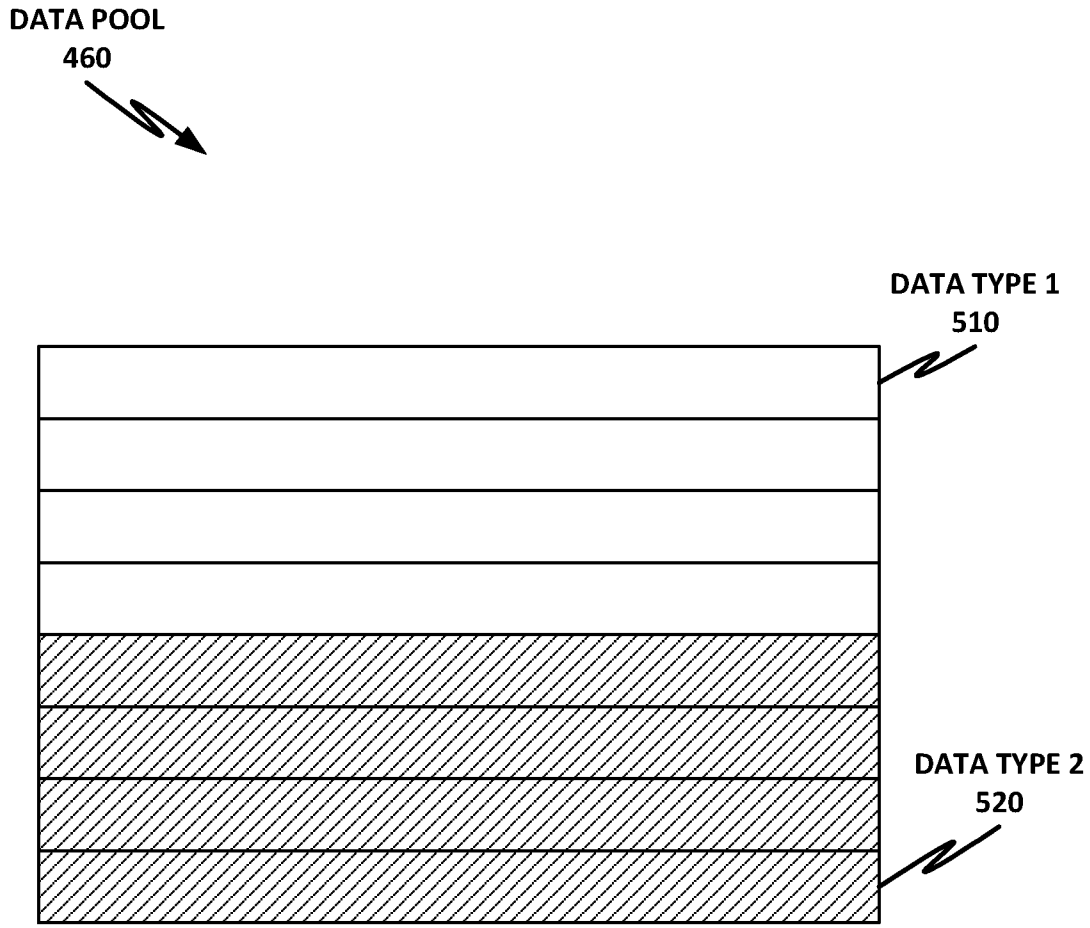


*Fig. 5*

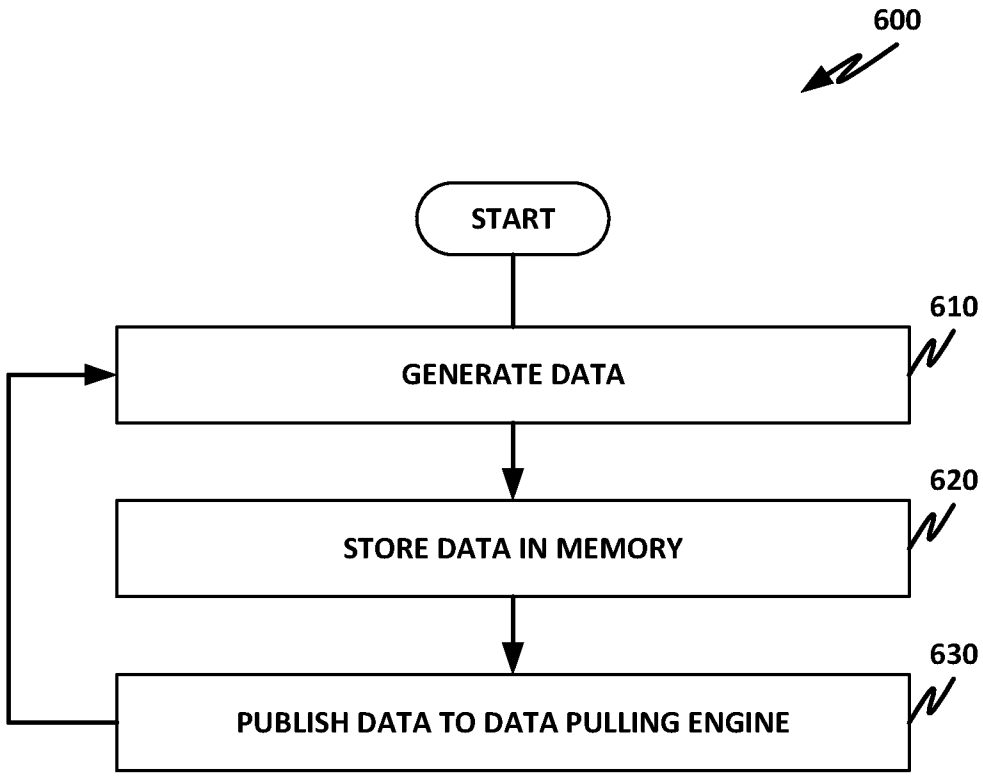
600
START
610
GENERATE DATA
620
STORE DATA IN MEMORY
630
PUBLISH DATA TO DATA PULLING ENGINE

*Fig. 6*

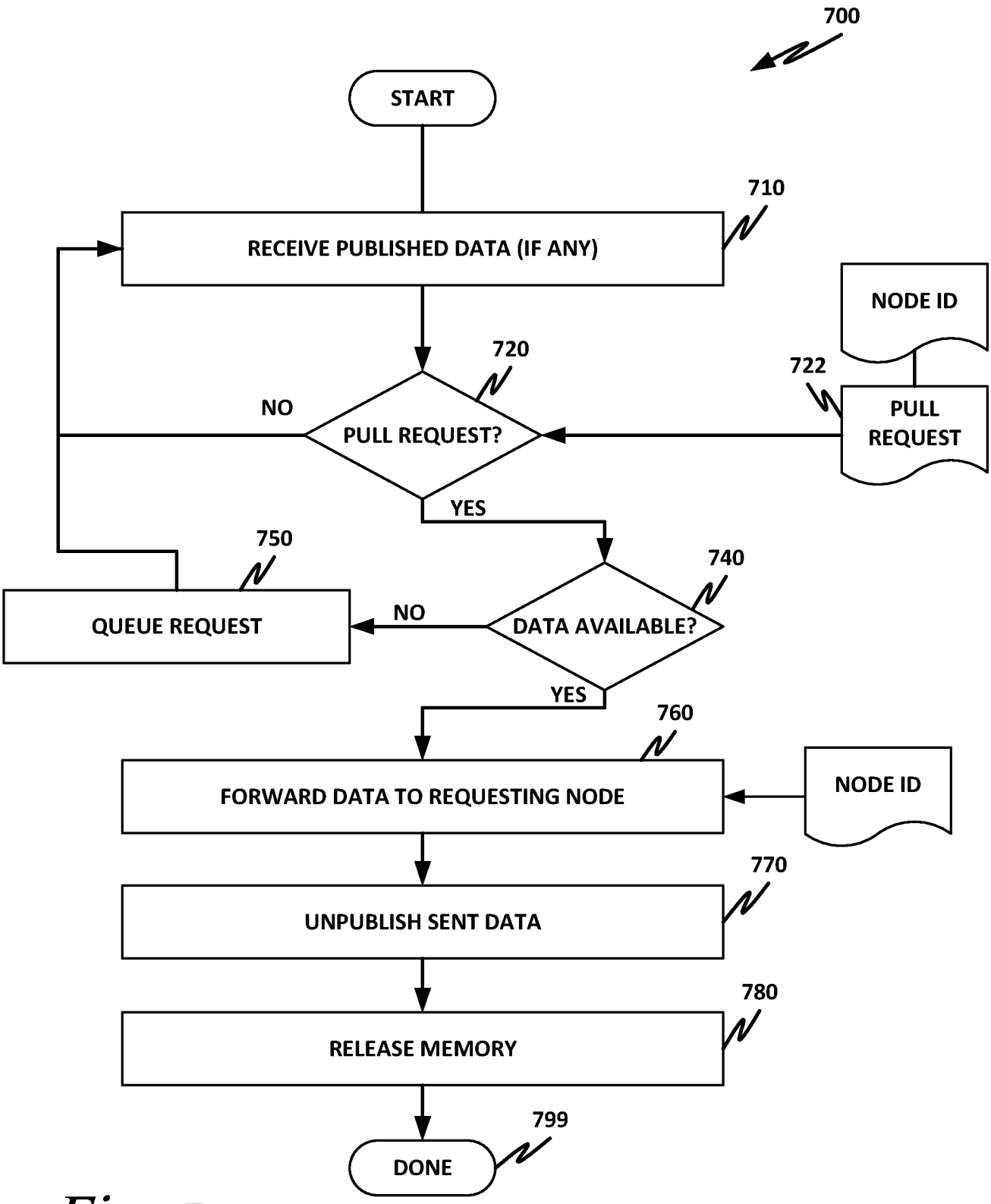
700
START
710
RECEIVE PUBLISHED DATA (IF ANY)
NODE ID
720
722
NO
PULL REQUEST?
PULL REQUEST
YES
750
740
QUEUE REQUEST
NO
DATA AVAILABLE?
YES
760
FORWARD DATA TO REQUESTING NODE
NODE ID
770
UNPUBLISH SENT DATA
780
RELEASE MEMORY
799
DONE

*Fig. 7*

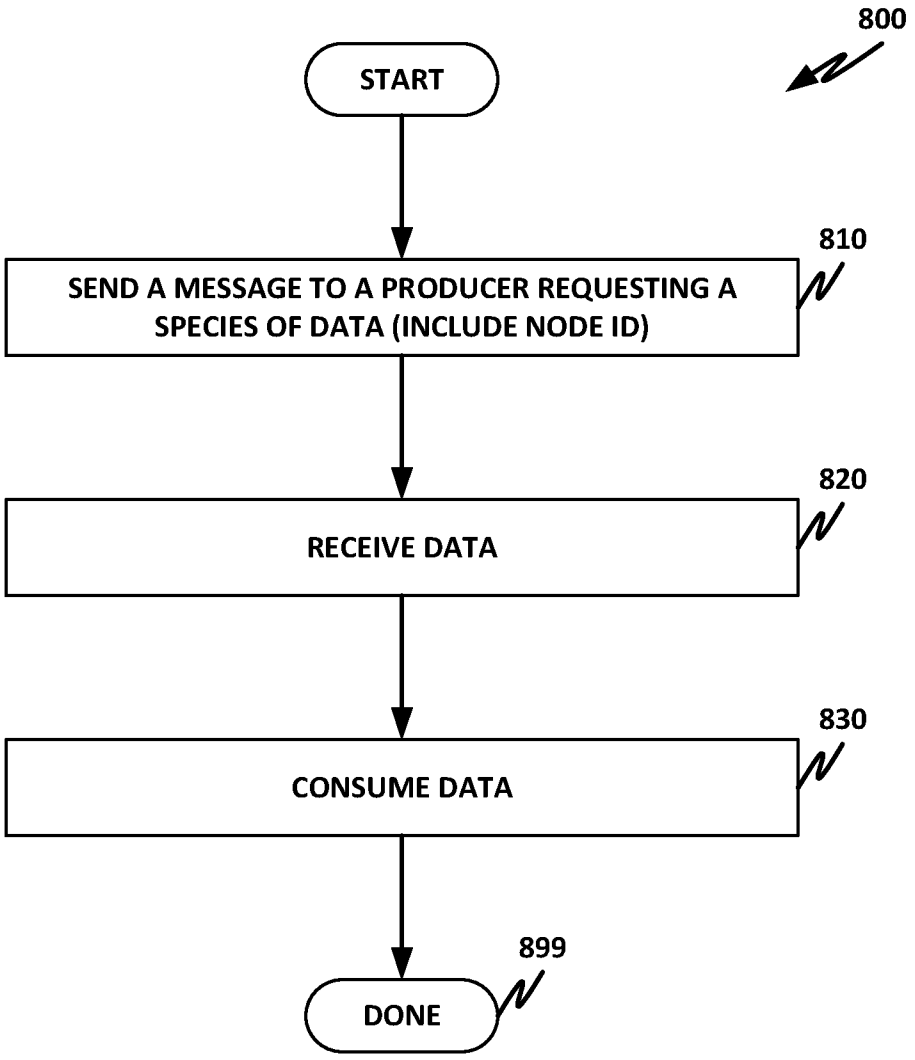
800
START
810
SEND A MESSAGE TO A PRODUCER REQUESTING A SPECIES OF DATA (INCLUDE NODE ID)
820
RECEIVE DATA
830
CONSUME DATA
899
DONE

*Fig. 8*

**INTERNATIONAL SEARCH REPORT**

International application No.
**PCT/US2016/063795**

**A. CLASSIFICATION OF SUBJECT MATTER**

**G06F 13/40(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 13/40; G06F 12/02; G06F 11/14; G06F 12/00; G06F 15/167; G06F 12/08

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: fabric, compute node, produce, datum, available, data pulling engine, pull request, node identifier, consumer, and similar terms.

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | US 2010-0306479 A1 (AHMED K. EZZAT) 02 December 2010<br>See paragraphs [0010], [0014], [0019]-[0020], [0023], and [0025]-[0030]; claim 12; and figures 1-2 and 5-6. | 1-4,9-17,22-25 |
| A | | 5-8,18-21,26-29 |
| Y | US 2010-0241813 A1 (XIAOWEI SHEN) 23 September 2010<br>See paragraphs [0011] and [0027]-[0028]; and figure 1. | 1-4,9-17,22-25 |
| A | US 2014-0173338 A1 (INTERNATIONAL BUSINESS MACHINES CORP.) 19 June 2014<br>See paragraphs [0025]-[0031] and figures 1-2. | 1-29 |
| A | US 2009-0198918 A1 (LAKSHMINARAYANA B. ARIMILLI et al.) 06 August 2009<br>See paragraphs [0041]-[0066] and figures 1-2. | 1-29 |
| A | WO 2006-055026 A1 (RAYTHEON COMPANY) 26 May 2006<br>See page 92, line 20 - page 94, line 9; and figure 13. | 1-29 |

☐ 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 |
|---|---|
| 28 April 2017 (28.04.2017) | **28 April 2017 (28.04.2017)** |


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
NHO, Ji Myong
Telephone No. +82-42-481-8528

**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2016/063795**

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| US 2010-0306479 A1 | 02/12/2010 | US 8024529 B2 | 20/09/2011 |
| US 2010-0241813 A1 | 23/09/2010 | US 7913048 B2 | 22/03/2011 |
| US 2014-0173338 A1 | 19/06/2014 | CN 104871493 A | 26/08/2015 |
| | | DE 112013006063 T5 | 03/09/2015 |
| | | GB 2523284 A | 19/08/2015 |
| | | GB 2523284 B | 07/10/2015 |
| | | JP 2016-509700 A | 31/03/2016 |
| | | US 9037898 B2 | 19/05/2015 |
| | | WO 2014-094521 A1 | 26/06/2014 |
| US 2009-0198918 A1 | 06/08/2009 | US 8484307 B2 | 09/07/2013 |
| WO 2006-055026 A1 | 26/05/2006 | EP 1815341 A1 | 08/08/2007 |
| | | JP 2008-521127 A | 19/06/2008 |
| | | JP 5570095 B2 | 13/08/2014 |
| | | US 2006-0112297 A1 | 25/05/2006 |
| | | US 7475274 B2 | 06/01/2009 |