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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2014 (13.03.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/039895 A1

(51) International Patent Classification:
H04L 29/08 (2006.01) *G06F 9/455* (2006.01)
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2013/058610

(22) International Filing Date:
6 September 2013 (06.09.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/698,216 7 September 2012 (07.09.2012) US
61/701,453 14 September 2012 (14.09.2012) US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5op7, Redwood Shores, CA 94065 (US).

(72) Inventors: **FALCO, Mark**; 74 Wilmington Road, Burlington, MA 01803 (US). **GLEYZER, Alex**; 35 Network Drive, Burlington, MA 01803 (US).

(74) Agents: **MEYER, Sheldon, R.** et al.; Fliesler Meyer LLP, 410 Pacific Avenue, San Francisco, CA 94133 (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, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

— *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))*

(54) Title: SYSTEM AND METHOD FOR SUPPORTING MESSAGE PRE-PROCESSING IN A DISTRIBUTED DATA GRID CLUSTER

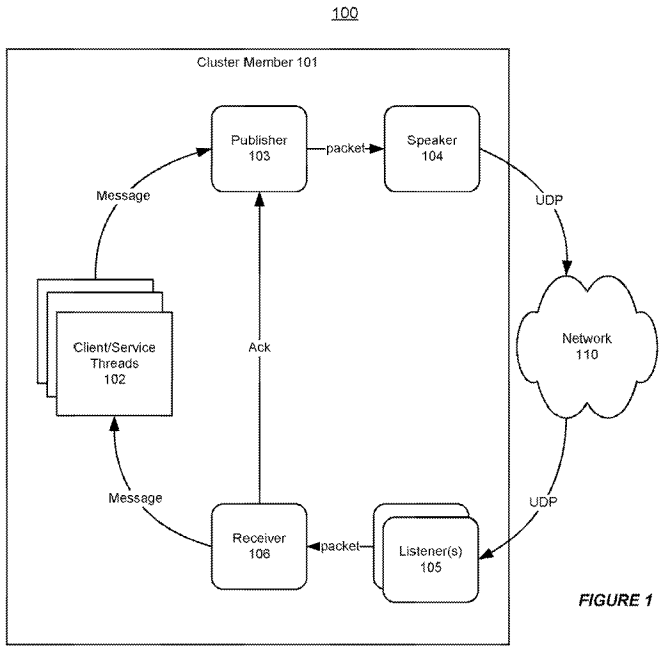

FIGURE 1

(57) Abstract: A system and method can support message pre-processing in a distributed data grid. The system can associate a message bus with a service thread on a cluster member in the distributed data grid. Furthermore, the system can receive one or more incoming messages at the message bus using an input/output (I/O) thread, and pre-process said one or more incoming messages on the I/O thread before each said incoming message is delivered to a service thread in the distributed data grid. Additionally, the system can take advantage of a pool of input/output (I/O) threads to deserialize inbound messages before they are delivered to the addressed service, and can relieve the bottleneck that is caused by performing all message deserialization in a single threaded fashion before the message type can be identified and offloaded to the thread-pool within the distributed data grid.

WO 2014/039895 A1

SYSTEM AND METHOD FOR SUPPORTING MESSAGE PRE-PROCESSING IN A DISTRIBUTED DATA GRID CLUSTER

Field of Invention:

[0002] The present invention is generally related to computer systems, and is particularly related to a distributed data grid.

Background:

[0003] Modern computing systems, particularly those employed by larger organizations and enterprises, continue to increase in size and complexity. In areas such as Internet applications, there is an expectation that millions of users should be able to simultaneously access that application, which effectively leads to an exponential increase in the amount of content generated and consumed by users, and transactions involving that content. Such activity also results in a corresponding increase in the number of transaction calls to databases and metadata stores, which have a limited capacity to accommodate that demand.

[0004] Furthermore, modern applications have embraced scale out architecture as a solution to the challenges of cost, scale and application reliability and serviceability. This approach offers many advantages over legacy approaches which are typically dependent on using increasingly large and costly high-end servers. However, this approach generally suffers from one persistent and challenging limitation: the input/output (I/O) bottleneck. Thus, the performance and efficiency of modern highly distributed systems may be constrained by the communication mechanism that connects all of the system components.

[0005] This is the general area that embodiments of the invention are intended to address.

Summary:

[0006] Described herein are systems and methods that can support message pre-processing in a distributed data grid. The system can associate a message bus with a service thread on a cluster member in the distributed data grid. Furthermore, the system can receive one or more incoming messages at the message bus using an input/output (I/O) thread, and pre-process said one or more incoming messages on the I/O thread before each said incoming message is delivered to a service thread in the distributed data grid. Additionally, the system can take advantage of a pool of input/output (IO) threads to deserialize inbound messages before they are

delivered to the addressed service, and can relieve the bottleneck that is caused by performing all message deserialization in a single threaded fashion before the message type can be identified and offloaded to the thread-pool within the distributed data grid.

[0007] An embodiment of the present invention provides an apparatus for supporting message pre-processing in a distributed data grid, comprising: means for associating a message bus with a service thread on a cluster member in the distributed data grid; means for receiving one or more incoming messages at the message bus using an input/output (I/O) thread; and means for pre-processing said one or more incoming messages on the I/O thread before each said incoming message is delivered to a said service thread in the distributed data grid.

[0008] In an example, the apparatus further comprises: means for executing, via the I/O thread, a pre-process method that is associated with a said incoming message.

[0009] In an example, the apparatus further comprises: means for deserializing each said incoming message using the I/O thread.

[00010] In an example, the apparatus further comprises: means for handling a said incoming message completely on the I/O thread and avoiding using the service thread.

[00011] In an example, the apparatus further comprises: means for processing said one or more pre-processed incoming messages on the service thread.

[00012] In an example, the apparatus further comprises: means for sending a response to a service requester that sends a said incoming message.

[00013] In an example, the apparatus further comprises: means for avoiding context switch that move said one or more incoming messages between the I/O thread and the service thread.

[00014] In an example, the apparatus further comprises: means for allowing the message bus to be based on a remote direct memory access (RDMA) protocol.

[00015] In an example, the apparatus further comprises: means for associating a thread pool with the message bus, wherein the thread pool contains a plurality of I/O threads.

[00016] In an example, the apparatus further comprises: means for processing said one or more incoming messages on the plurality of I/O threads in parallel.

[00017] Another embodiment of the present invention provides a system for message pre-processing in a distributed data grid, comprising: a first associating unit configured to associate a message bus with a service thread on a cluster member in the distributed data grid; a receiving unit configured to receive one or more incoming messages at the message bus using an input/output (I/O) thread; and a pre-processing unit configured to pre-process said one or more incoming messages on the I/O thread before each said incoming message is delivered to a said service thread in the distributed data grid

[00018] In an example, the system further comprises: an executing unit configured to execute, via the I/O thread, a pre-process method that is associated with a said incoming message.

[00019] In an example, the system further comprises: a deserializing unit configured to deserialize each said incoming message using the I/O thread.

[00020] In an example, the system further comprises: a handling unit configured to handle a said incoming message completely on the I/O thread and avoid using the service thread.

[00021] In an example, the system further comprises: a first processing unit configured to process said one or more pre-processed incoming messages on the service thread.

[00022] In an example, the system further comprises: a sending unit configured to send a response to a service requester that sends a said incoming message.

[00023] In an example, the system further comprises: a context switch avoiding unit configured to avoid context switch that move said one or more incoming messages between the I/O thread and the service thread.

[00024] In an example, the message bus is based on a remote direct memory access (RDMA) protocol.

[00025] In an example, the system further comprises: a second associating unit configured to associate a thread pool with the message bus, wherein the thread pool contains a plurality of I/O threads.

[00026] In an example, the system further comprises: a second processing unit configured to process said one or more incoming messages on the plurality of I/O threads in parallel.

Brief Description of the Figures:

[00027] **Figure 1** shows an illustration of supporting message transport based on a datagram layer in a distributed data grid.

[00028] **Figure 2** shows an illustration of providing a message bus in a distributed data grid, in accordance with an embodiment of the invention.

[00029] **Figure 3** shows an illustration of using a TCP/IP based transport layer to support messaging in a distributed data grid.

[00030] **Figure 4** shows an illustration of using a RDMA based transport layer to support messaging in a distributed data grid, in accordance with an embodiment of the invention.

[00031] **Figure 5** shows an illustration of supporting bus per service in a distributed data grid, in accordance with an embodiment of the invention.

[00032] **Figure 6** illustrates an exemplary flow chart for supporting bus per service in a distributed data grid, in accordance with an embodiment of the invention.

[00033] **Figure 7** shows an illustration of supporting parallel message deserialization in a distributed data grid, in accordance with an embodiment of the invention.

[00034] **Figure 8** illustrates an exemplary flow chart for supporting parallel message deserialization in a distributed data grid, in accordance with an embodiment of the invention.

[00035] **Figure 9** shows an illustration of supporting message pre-processing in a distributed data grid, in accordance with an embodiment of the invention.

[00036] **Figure 10** illustrates an exemplary flow chart for supporting message pre-processing in a distributed data grid, in accordance with an embodiment of the invention.

[00037] **Figure 11** illustrates a schematic functional block diagram of a system in accordance with an embodiment of the invention.

[00038] **Figure 12** illustrates a functional block diagram of a system for message pre-processing in a distributed data grid in accordance with an embodiment of the present invention.

Detailed Description:

[00039] The invention is illustrated, by way of example and not by way of limitation, in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" or "some" embodiment(s) in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[00040] The description of the embodiments of the invention as following uses a Coherence distributed data grid as an example for a distributed data grid. It will be apparent to those skilled in the art that other types distributed data grids can be used without limitation. Furthermore, the description of the invention as following uses an Exabus messaging mechanism as an example for a messaging mechanism. It will be apparent to those skilled in the art that other types messaging mechanisms can be used without limitation.

[00041] Described herein are systems and methods that can support a scalable message bus in a distributed data grid cluster. The scalable message bus can provide each service with its own bus (transport engine). The distributed data grid can take advantage of a pool of input/output (I/O) threads to deserialize inbound messages before they are delivered to the addressed service, and can relieve the bottleneck that is caused by performing all message deserialization in a single threaded fashion before the message type can be identified and offloaded to the thread-pool within the distributed data grid. Additionally, the distributed data grid allows incoming messages to be pre-processed on the I/O thread for the scalable message bus.

Distribute Data Grid

[00042] In accordance with an embodiment, as referred to herein a "data grid cluster", or "data grid", is a system comprising a plurality of computer servers which work together to manage information and related operations, such as computations, within a distributed or clustered environment. The data grid cluster can be used to manage application objects and data that are shared across the servers. Preferably, a data grid cluster should have low response time, high throughput, predictable scalability, continuous availability and information reliability. As a result of these capabilities, data grid clusters are well suited for use in computational intensive, stateful middle-tier applications. Some examples of data grid clusters, e.g., the Oracle Coherence data grid cluster, can store the information in-memory to achieve higher performance, and can employ redundancy in keeping copies of that information synchronized across multiple servers, thus ensuring resiliency of the system and the availability of the data in the event of server failure. For

example, Coherence provides replicated and distributed (partitioned) data management and caching services on top of a reliable, highly scalable peer-to-peer clustering protocol.

[00043] An in-memory data grid can provide the data storage and management capabilities by distributing data over a number of servers working together. The data grid can be middleware that runs in the same tier as an application server or within an application server. It can provide management and processing of data and can also push the processing to where the data is located in the grid. In addition, the in-memory data grid can eliminate single points of failure by automatically and transparently failing over and redistributing its clustered data management services when a server becomes inoperative or is disconnected from the network. When a new server is added, or when a failed server is restarted, it can automatically join the cluster and services can be failed back over to it, transparently redistributing the cluster load. The data grid can also include network-level fault tolerance features and transparent soft re-start capability.

[00044] In accordance with an embodiment, the functionality of a data grid cluster is based on using different cluster services. The cluster services can include root cluster services, partitioned cache services, and proxy services. Within the data grid cluster, each cluster node can participate in a number of cluster services, both in terms of providing and consuming the cluster services. Each cluster service has a service name that uniquely identifies the service within the data grid cluster, and a service type, which defines what the cluster service can do. Other than the root cluster service running on each cluster node in the data grid cluster, there may be multiple named instances of each service type. The services can be either configured by the user, or provided by the data grid cluster as a default set of services.

[00045] **Figure 1** shows an illustration of supporting message transport based on a datagram layer in a distributed data grid. As shown in Figure 1, a cluster member 101 in a distributed data grid 100 can include one or more client/service threads 102. The client/service threads 102 on the cluster member 101 can send a message to other cluster members in the distributed data grid 100 through a network, e.g. an Ethernet network 110, using a user datagram protocol (UDP).

[00046] In accordance with an embodiment of the invention, the cluster member 101 can employ different logics, such as packetization logic, packet retransmission logic, and Ack/Nack logic, for sending a message to another cluster member in the distributed data grid 100 and receiving a response message.

[00047] Additionally, the above messaging process can involve multiple context switches. As shown in Figure 1, the client/service thread 102 can first send the message to a publisher 103. Then, the publisher 103 can forward the message to a speaker 104, which is responsible for sending the message to the network 110.

[00048] Furthermore, the cluster member 101 in a distributed data grid 100 can receive a response message using one or more listeners 105, which can forward the received message to a receiver 106. Then, the receiver 106 can forward the received message to the client/service thread 102 and, optionally, notify the publisher 103.

Scalable Message Bus

[00049] In accordance with an embodiment of the invention, a scalable message bus can be used for eliminating I/O bottlenecks at various levels.

[00050] **Figure 2** shows an illustration of providing a message bus in a distributed data grid, in accordance with an embodiment of the invention. As shown in Figure 2, a cluster member 201 can run on a virtual machine 210, e.g. a JAVA virtual machine, in a distributed data grid 200. The cluster member 201 can involve one or more services 211, which can use one or more message buses 212 for messaging.

[00051] In accordance with an embodiment of the invention, the message buses 212 can be based on a binary low-level message transport layer, with multi-point addressing and reliable ordered delivery. Also, the message bus can be based on pure Java implementation and/or native implementations, and can employ an asynchronous event based programming model.

[00052] Furthermore, the message bus 212 can be supported using a networking hardware and software subsystem, e.g. an Exabus in Oracle ExaLogic engineered system. The message bus can not only make applications running faster, and can also make the applications running more efficiently. Moreover, and the message bus can make applications running consistently and predictably, even in extremely large scale deployments with thousands of processor cores and terabytes of memory and for virtually all business applications.

[00053] In accordance with an embodiment of the invention, each of the message buses 212 can be a provider-based transport layer, which can be supported by using a message bus provider 202 in the virtual machine 210, such as JRockit/HotSpot.

[00054] Additionally, the message bus provider 202 can be based on a pluggable provider based framework. For example, the message bus provider 202 can support different message buses such as a SocketBus, which is based on TCP/SDP, and an InfiniBus, which is based on Infiniband RDMA.

[00055] In accordance with an embodiment of the invention, the message bus provider can use a single switch to select from a bus protocol from a plurality of bus protocols. For example, in Coherence, the system can specify the single switch in the following configuration file

```
Dtangosol.coherence.transport.reliable=protocol
    <cluster-config>!
        <unicast-listener>!
        <reliable-transport>protocol</reliable-transport>
```

[00056] Additionally, in Coherence, the system can use the single switch to select one of the following buses, such as

- tmb: TCP MessageBus
- sdmb: SDP MessageBus
- imb: Infiniband MessageBus
- datagram: legacy UDP (default)

[00057] Thus, the message buses 212 can improve intra-node scalability in the distributed data grid 200, and can make the distributed data grid 200 protocol agnostic. For example, using the message buses 212, the distributed data grid 200 can effectively utilize large number of cores, improve messaging concurrency, and increase throughput and reduce latency. Also, the message buses 212 allow the distributed data grid 200 to minimize context switches and take advantage of the zero copy.

[00058] In accordance with an embodiment of the invention, the system can trigger death detection on the cluster member when a message bus fails.

[00059] **Figure 3** shows an illustration of using a TCP/IP based transport layer to support messaging in a distributed data grid. As shown in Figure 3, in order to send a message from application 301 to application 302 in a distributed computing environment 300, the message may need to go through an application buffer 303, a TCP/IP transport layer 305 and a kernel layer 306 on a local machine. Then, the message can be received at remote machine in an application buffer 304, via the kernel layer 306 and the TCP/IP transport layer 305 in the remote machine.

[00060] **Figure 4** shows an illustration of using a RDMA based transport layer to support messaging in a distributed data grid. As shown in Figure 4, the system can send a message from an application 401 on a local machine directly to an application 401 on a remote machine, based on the RDMA based transport layer.

Bus per Service

[00061] In accordance with an embodiment of the invention, a scalable message bus can provide each service with its own bus (or transport engine).

[00062] **Figure 5** shows an illustration of supporting a scalable message bus for various services in a distributed data grid, in accordance with an embodiment of the invention. As shown in Figure 5, a distributed data grid 500 can include multiple cluster members, e.g. cluster members 501-504.

[00063] Furthermore, each cluster member can include different services, each of which can be associated with a separate message bus. For example, the cluster member 501 can include partition cache services 511-512 and invocation service 513, which can be associated with message buses 514-516; the cluster member 502 can include partition cache services 521-522 and invocation service 523, which can be associated with message buses 524-526; the cluster member 503 can include partition cache services 531-532 and invocation service 533, which can be associated with message buses 534-536; and the cluster member 504 can include partition

cache services 541-542 and invocation service 543, which can be associated with message buses 544-546.

[00064] Additionally, a network 510 can connect different message buses on different cluster members 501-504 in the distributed data grid 500. For example, the network 510 can be based on a remote direct memory access (RDMA) protocol. Moreover, the network510 can fall back on a user datagram protocol (UDP) if necessary.

[00065] In accordance with an embodiment of the invention, the system can use the plurality of message buses to support data transferring between different cluster members in the distributed data grid. Additionally, the system can use a datagram layer 520 to support clustering in the distributed data grid, and can bypass the datagram layer 520 in the distributed data grid for data transferring.

[00066] Thus, the system allows an increase in CPU utilization relative to the number of services configured by the end user. Unlike in a traditional networking model, a single transport engine can be provided per service instead of per cluster node, such that the distributed data grid can relieve the bottleneck when too many processors try to utilize a single cluster node.

[00067] **Figure 6** illustrates an exemplary flow chart for supporting message transport based on a provider-based transport layer in a distributed data grid, in accordance with an embodiment of the invention. As shown in Figure 6, at step 601, the system can provide a plurality of message buses in the distributed data grid, wherein the distributed data grid includes a plurality of cluster members. Furthermore, at step 602, the system can associate each service in the distributed data grid with a said message bus, and, at step 603, the system can use the plurality of message buses to support data transferring between different cluster members in the distributed data grid.

Parallel Message Deserialization

[00068] In accordance with an embodiment of the invention, a pool of threads can be used to provide threads, such as input/output (I/O) threads, for driving a scalable message bus to handle inbound messages in a distributed data grid, e.g. a Coherence data grid. Furthermore, by offloading the messages to a thread pool within the distributed data grid and deserialize the message up front on the I/O thread, the system can minimize the impact of the service thread bottleneck.

[00069] **Figure 7** shows an illustration of supporting parallel message deserialization in a distributed data grid, in accordance with an embodiment of the invention. As shown in Figure 7, a service thread 702 in a distributed data grid 700 can be associated with a message bus 701, which can receive one or more incoming messages, e.g. messages 703-704.

[00070] Furthermore, the message bus 701 can be associated with a thread pool 710, which contains one or more threads, e.g. I/O threads 711-713. The distributed data grid 700 can take advantage of this thread pool 710 to relieve the performance bottleneck at the service thread 702.

[00071] In accordance with an embodiment of the invention, the distributed data grid 700 can

use multiple different I/O threads 711-713 in the thread pool 710 to process the incoming messages 703-704 in parallel. Thus, the system can avoid the service thread bottleneck caused by performing all message deserialization in a single threaded before the message type can be identified.

[00072] For example, when the direct memory access (RDMA) protocol is used, the message bus 701 can use the I/O thread 711 to deserialize the message 703, before delivering the incoming message 703 to the service thread 702. Also, the message bus 701 can use the I/O thread 713 to deserialize the message 704, before delivering the incoming message 704 to the service thread 702.

[00073] **Figure 8** illustrates an exemplary flow chart for supporting parallel message deserialization in a distributed data grid, in accordance with an embodiment of the invention. As shown in Figure 8, at step 801, the system can provide a pool of threads to provide a plurality of input/output (I/O) threads that operates to drive a scalable message bus. Furthermore, at step 802, the system can receive one or more inbound messages on the plurality of IO threads, and, at step 803, the system can deserialize the one or more inbound messages on the plurality of I/O threads before delivering the one or more inbound messages to the addressed service.

Message pre-processing

[00074] In accordance with an embodiment of the invention, a scalable message bus can provide message pre-processing capability, which allows pre-processing the received messages, e.g. on the input/output (I/O) thread, before delivering the received messages to a service thread.

[00075] **Figure 9** shows an illustration of supporting message pre-processing in a distributed data grid, in accordance with an embodiment of the invention. As shown in Figure 9, a service thread 902 in the distributed data grid 900 can be associated with a message bus 901.

[00076] The message bus 901 can use one or more I/O threads, e.g. an I/O thread 903, to receive one or more incoming messages, e.g. a message 905. Additionally, the message bus 901 can use the I/O thread 903 to deserialize the incoming message 905.

[00077] Once the incoming message 905 is deserialized, the message bus 901 can pre-process the incoming message 905, before delivering it to the service thread 902. Then, the service thread 902 can further complete processing the pre-processed incoming messages 905, and, if necessary, can send a response 907 back to the service requester that sends the incoming message 905.

[00078] In accordance with an embodiment of the invention, the incoming message 905 can provide a pre-process method 906. The message bus 901 can execute the pre-process method 906 associated with the incoming message 905 on the I/O thread 903, in order to partially or fully process the incoming message 905. Furthermore, for various message types, it may be possible to completely handle the message execution (reaching the end/response 907 during the

pre-processing phase) and avoid the service thread entirely.

[00079] In accordance with an embodiment of the invention, the message pre-processing capability of the scalable message bus can be beneficial when it is used in the distributed data grid 900. First, by taking advantage of the message pre-processing capability, the system can avoid overburden the service thread, since the service thread can be a bottleneck in the distributed data grid. Second, using message pre-processing, the system avoids the context switches that may be required when moving the message between the I/O thread 903 and the service thread 902. Such context switches can cause a significant percentage of the overall request latency, e.g. in the case of a remote direct memory access (RDMA) based transport. Third, the scalable message bus allows the messages to be fully executed in parallel if the scalable message bus has multiple IO threads such as in the case of a RDMA based bus.

[00080] Furthermore, the scalable message bus can combine the message pre-processing capability with the parallel message deserialization capability, so that multiple incoming messages can be deserialized and pre-processed in parallel.

[00081] **Figure 10** illustrates an exemplary flow chart for supporting message pre-processing in a distributed data grid, in accordance with an embodiment of the invention. As shown in Figure 10, at step 1001, the system can associate a message bus with a service thread on a cluster member in the distributed data grid. Furthermore, at step 1002, the system can receive one or more incoming messages at the message bus using an input/output (I/O) thread, and, at step 1003, the system can pre-process said one or more incoming messages on the I/O thread before each said incoming message is delivered to a service thread in the distributed data grid.

[00082] **Figure 11** illustrates a schematic functional block diagram of a system 1100 for message pre-processing in a distributed data grid in accordance with an embodiment of the invention. System 1100 includes: one or more microprocessors and a cluster member in the distributed data grid running on the one or more microprocessors. System 1100 includes, in the cluster member, an associattion module 1110, a message bus 1120, a service thread 1130, and a pre-processing module 1140. Association module 1110 is adapted for associating message bus 1120 with service thread 1130 on a cluster member in the distributed data grid. Message bus 1120 is adapted for receiving one or more incoming messages using an input/output (I/O) thread. Pre-processing module 1140 is adapted for pre-process the one or more incoming messages on the I/O thread before each incoming message is delivered to service thread 1130 in the data grid.

[00083] **Figure 12** illustrates a functional block diagram of a system 1200 for message pre-processing in a distributed data grid in accordance with the principles of the present invention as described above. The functional blocks of the system 1200 may be implemented by hardware, software, or a combination of hardware and software to carry out the principles of the present invention. It is understood by those skilled in the art that the functional blocks described in Figure 12 may be combined or separated into sub-blocks to implement the principles of the present invention as described above. Therefore, the description herein may support any possible

combination or separation or further definition of the functional blocks described herein.

[00084] As shown in Figure 12, the system 1200 for message pre-processing in a distributed data grid comprises a first associating unit 1202, a receiving unit 1204 and a pre-processing unit 1206. The first associating unit 1202 is configured to associate a message bus with a service thread on a cluster member in the distributed data grid. The receiving unit 1204 is configured to receive one or more incoming messages at the message bus using an input/output (I/O) thread. The pre-processing unit 1106 is configured to pre-process said one or more incoming messages on the I/O thread before each said incoming message is delivered to a said service thread in the distributed data grid.

[00085] In an example, the system 1200 further comprises an executing unit 1208 that is configured to execute, via the I/O thread, a pre-process method that is associated with a said incoming message.

[00086] In an example, the system 1200 further comprises a deserializing unit 1210 that is configured to deserialize each said incoming message using the I/O thread.

[00087] In an example, the system 1100 further comprises a handling unit 1212 that is configured to handle a said incoming message completely on the I/O thread and avoid using the service thread.

[00088] In an example, the system 1200 further comprises a first processing unit 1214 that is configured to process said one or more pre-processed incoming messages on the service thread.

[00089] In an example, the system 1200 further comprises a sending unit 1216 that is configured to send a response to a service requester that sends a said incoming message.

[00090] In an example, the system 1200 further comprises a context switch avoiding unit 1218 that is configured to avoid context switch that move said one or more incoming messages between the I/O thread and the service thread.

[00091] In an example, the message bus is based on a remote direct memory access (RDMA) protocol.

[00092] In an example, the system 1200 further comprises a second associating unit 1220 configured to associate a thread pool with the message bus, wherein the thread pool contains a plurality of I/O threads.

[00093] In an example, the system 1200 further comprises a second processing unit 1222 that is configured to process said one or more incoming messages on the plurality of I/O threads in parallel.

[00094] The present invention may be conveniently implemented using one or more conventional general purpose or specialized digital computer, computing device, machine, or microprocessor, including one or more processors, memory and/or computer readable storage media programmed according to the teachings of the present disclosure. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.

[00095] In some embodiments, the present invention includes a computer program product which is a storage medium or computer readable medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[00096] The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. The modifications and variations include any relevant combinations of the disclosed features. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

Claims:

What is claimed is:

1. A method for supporting message pre-processing in a distributed data grid operating on one or more microprocessors, comprising:

associating a message bus with a service thread on a cluster member in the distributed data grid;

receiving one or more incoming messages at the message bus using an input/output (I/O) thread; and

pre-processing said one or more incoming messages on the I/O thread before each said incoming message is delivered to said service thread in the distributed data grid.

2. The method according to Claim 1, further comprising:

executing, via the I/O thread, a pre-process method that is associated with an incoming message.

3. The method according to Claim 1 or 2, further comprising:

deserializing each said incoming message using the I/O thread.

4. The method according to any of Claims 1 to 3, further comprising:

handling an incoming message completely on the I/O thread and avoiding using the service thread.

5. The method according to any of Claims 1 to 4, further comprising:

processing said one or more pre-processed incoming messages on the service thread.

6. The method according to any of Claims 1 to 5, further comprising:

sending a response to a service requester that sends a said incoming message.

7. The method according to any of Claims 1 to 6, further comprising:

avoiding context switch that would move said one or more incoming messages between the I/O thread and the service thread.

8. The method according to any of Claims 1 to 7, further comprising:

allowing the message bus to be based on a remote direct memory access (RDMA) protocol.

9. The method according to any of Claims 1 to 8, further comprising:

associating a thread pool with the message bus, wherein the thread pool contains a plurality of I/O threads.

10. The method according to Claim 9, further comprising:

processing said one or more incoming messages on the plurality of I/O threads in parallel.

11. A computer program comprising machine readable instructions which when executed by one or more computer systems cause the one or more computer systems to perform the method of any preceding claim.

12. A system for message pre-processing in a distributed data grid, comprising:

one or more microprocessors,

a cluster member in the distributed data grid running on the one or more microprocessors, wherein the cluster member operate to

associate a message bus with a service thread on a cluster member in the distributed data grid;

receive one or more incoming messages at the message bus using an input/output (I/O) thread; and

pre-process said one or more incoming messages on the I/O thread before each said incoming message is delivered to a said service thread in the distributed data grid

13. The system according to Claim 12, wherein:

the cluster member operate to execute, via the I/O thread, a pre-process method that is associated with a said incoming message.

14. The system according to Claim 12 or 13, wherein:

the cluster member operate to deserialize each said incoming message using the I/O thread.

15. The system according to any of Claims 12 to 14, wherein:

the cluster member operate to handle a said incoming message completely on the I/O thread and avoid using the service thread.

16. The system according to any of Claims 12 to 14, wherein:

the cluster member operate to process said one or more pre-processed incoming messages on the service thread.

17. The system according to any of Claims 12 to 16, wherein:

the cluster member operate to send a response to a service requester that sends a said incoming message.

18. The system according to any of Claims 12 to 17, wherein:

the cluster member operate to avoid context switch that move said one or more incoming messages between the I/O thread and the service thread.

19. The system according to any of Claims 12 to 18, wherein:

the message bus is based on a remote direct memory access (RDMA) protocol.

20. The system according to any of Claims 12 to 19, wherein:

the cluster member operate to

associate a thread pool with the message bus, wherein the thread pool contains a plurality of I/O threads; and

process said one or more incoming messages on the plurality of I/O threads in parallel.

21. A non-transitory machine readable storage medium having instructions stored thereon that when executed cause a system to perform the steps comprising:

associating a message bus with a service thread on a cluster member in a distributed data grid;

receiving one or more incoming messages at the message bus using an input/output (I/O) thread; and

pre-processing said one or more incoming messages on the I/O thread before each said incoming message is delivered to a said service thread in the distributed data grid.

22. A computer program product comprising a machine readable storage medium storing the computer program of Claim 11.

23. A program for causing one or more computers to implement the method recited in any one of Claims 1 to 10.

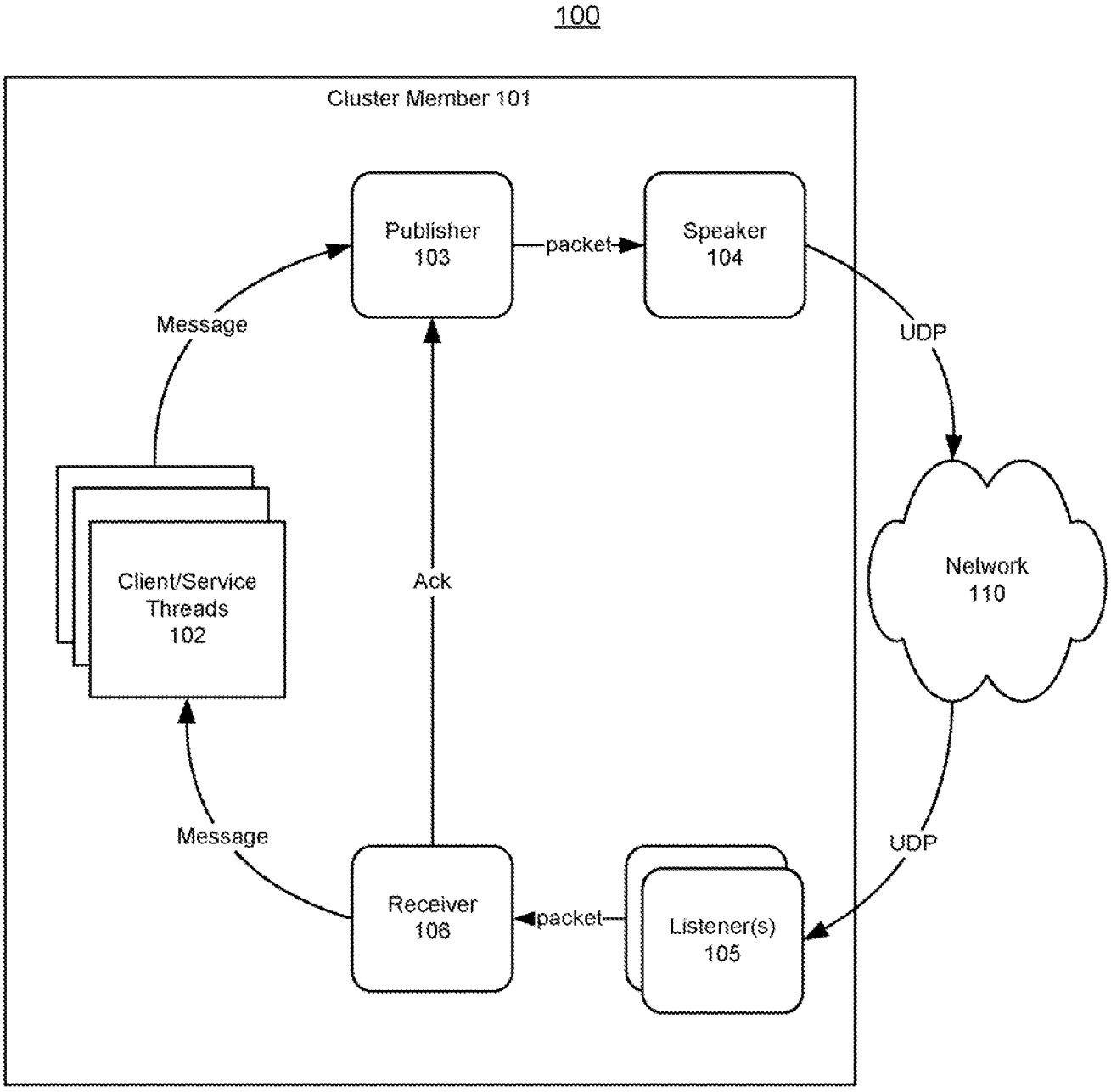
100
Cluster Member 101
Publisher
103
packet
Speaker
104
Message
UDP
Client/Service
Threads
102
Ack
Network
110
Message
UDP
Receiver
106
packet
Listener(s)
105

FIGURE 1

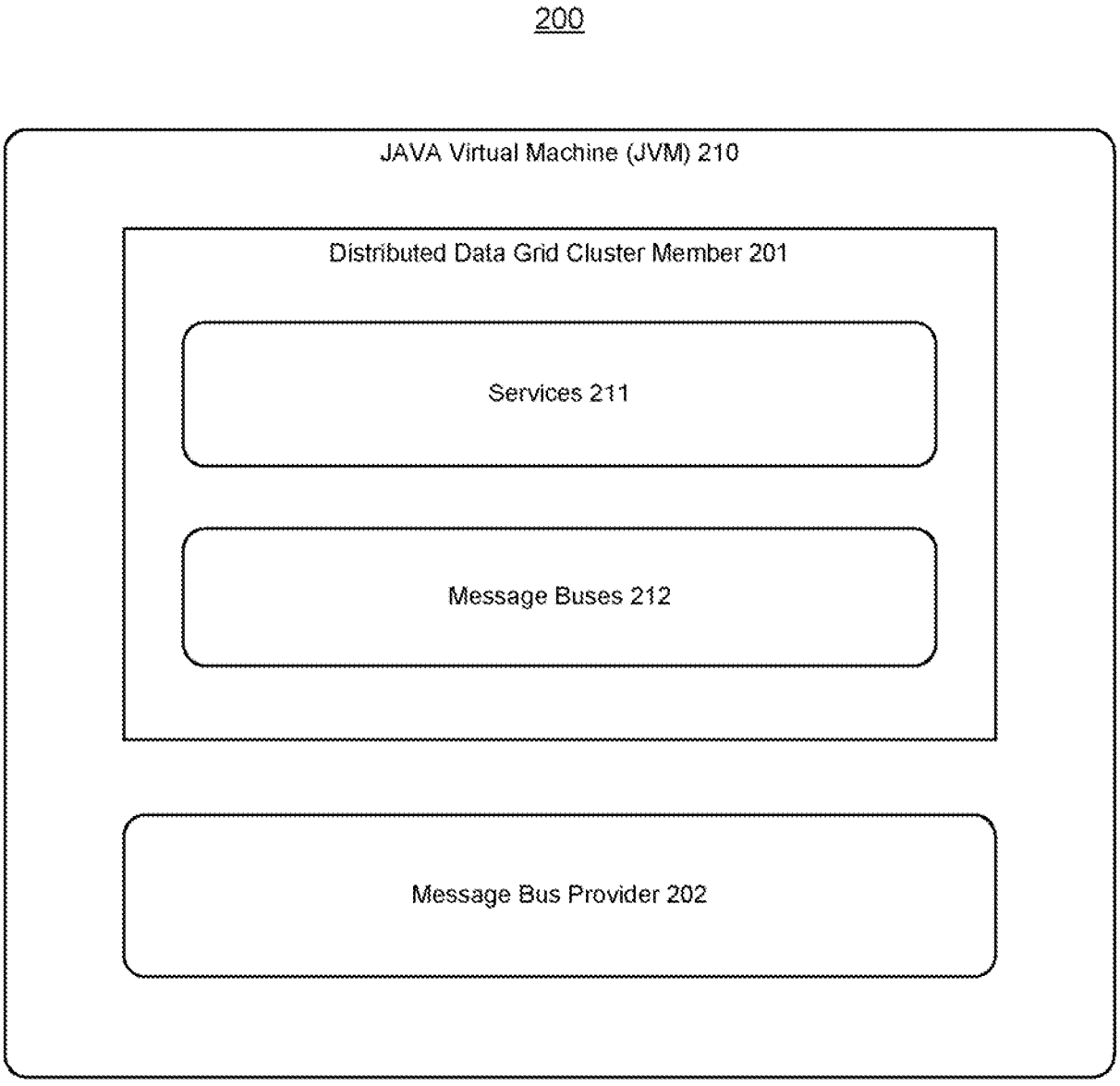
200
JAVA Virtual Machine (JVM) 210
Distributed Data Grid Cluster Member 201
Services 211
Message Buses 212
Message Bus Provider 202

FIGURE 2

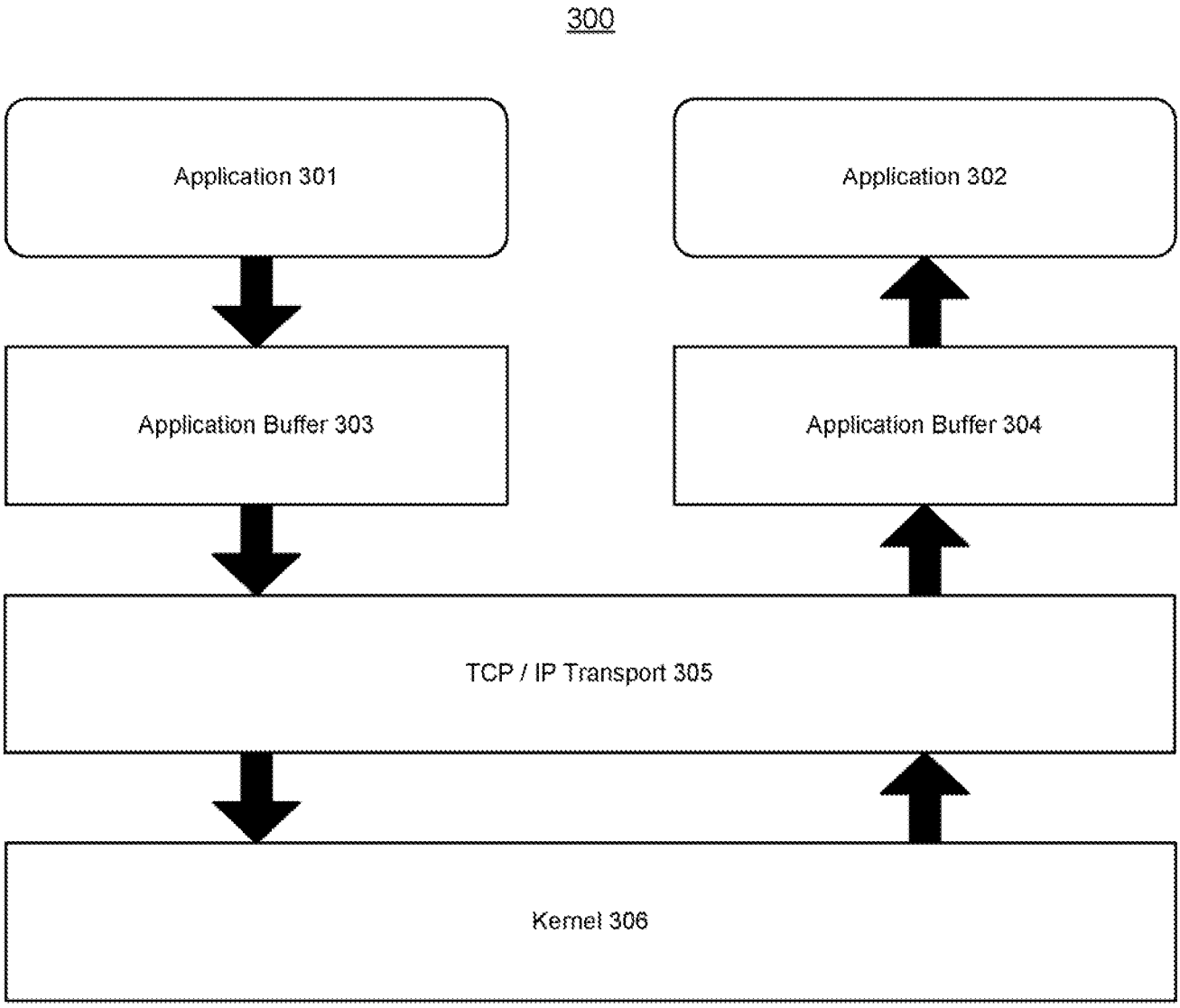
300
Application 301
Application 302
Application Buffer 303
Application Buffer 304
TCP / IP Transport 305
Kernel 306

FIGURE 3

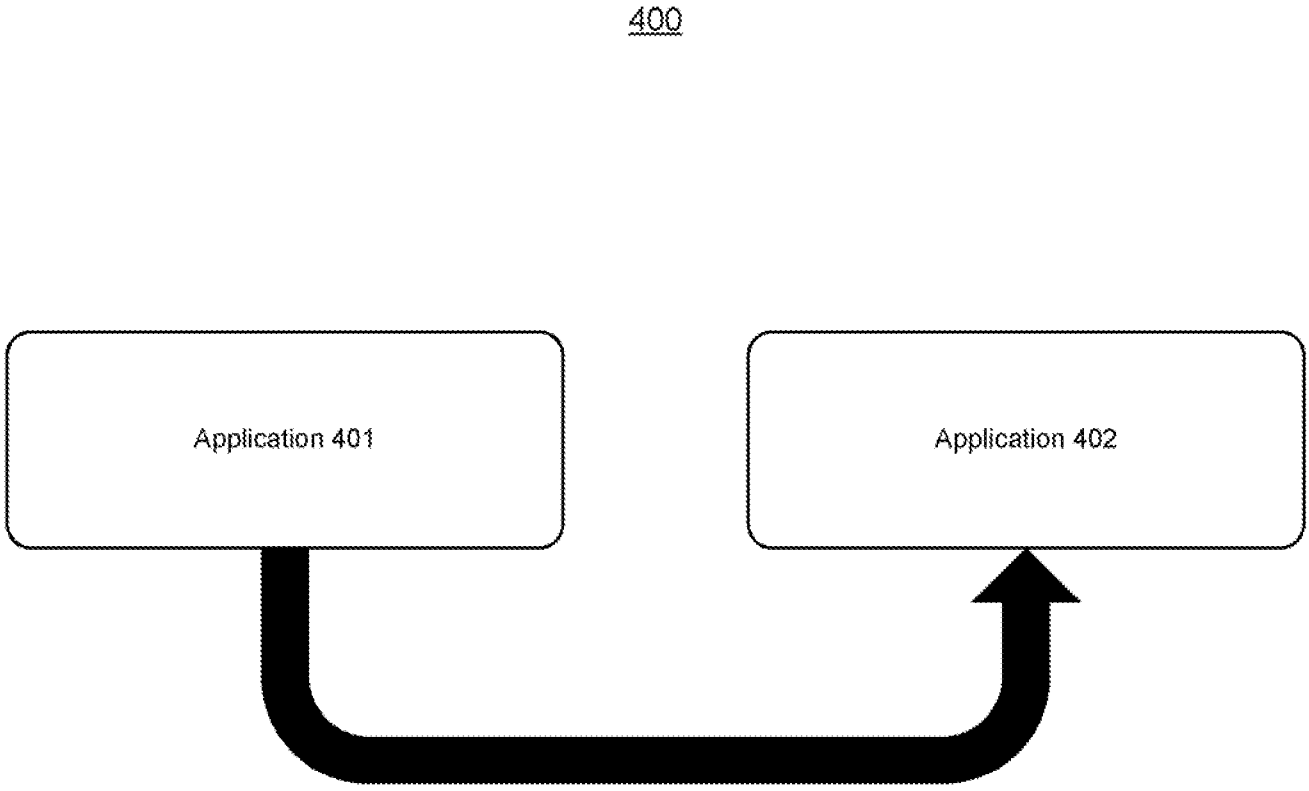

FIGURE 4

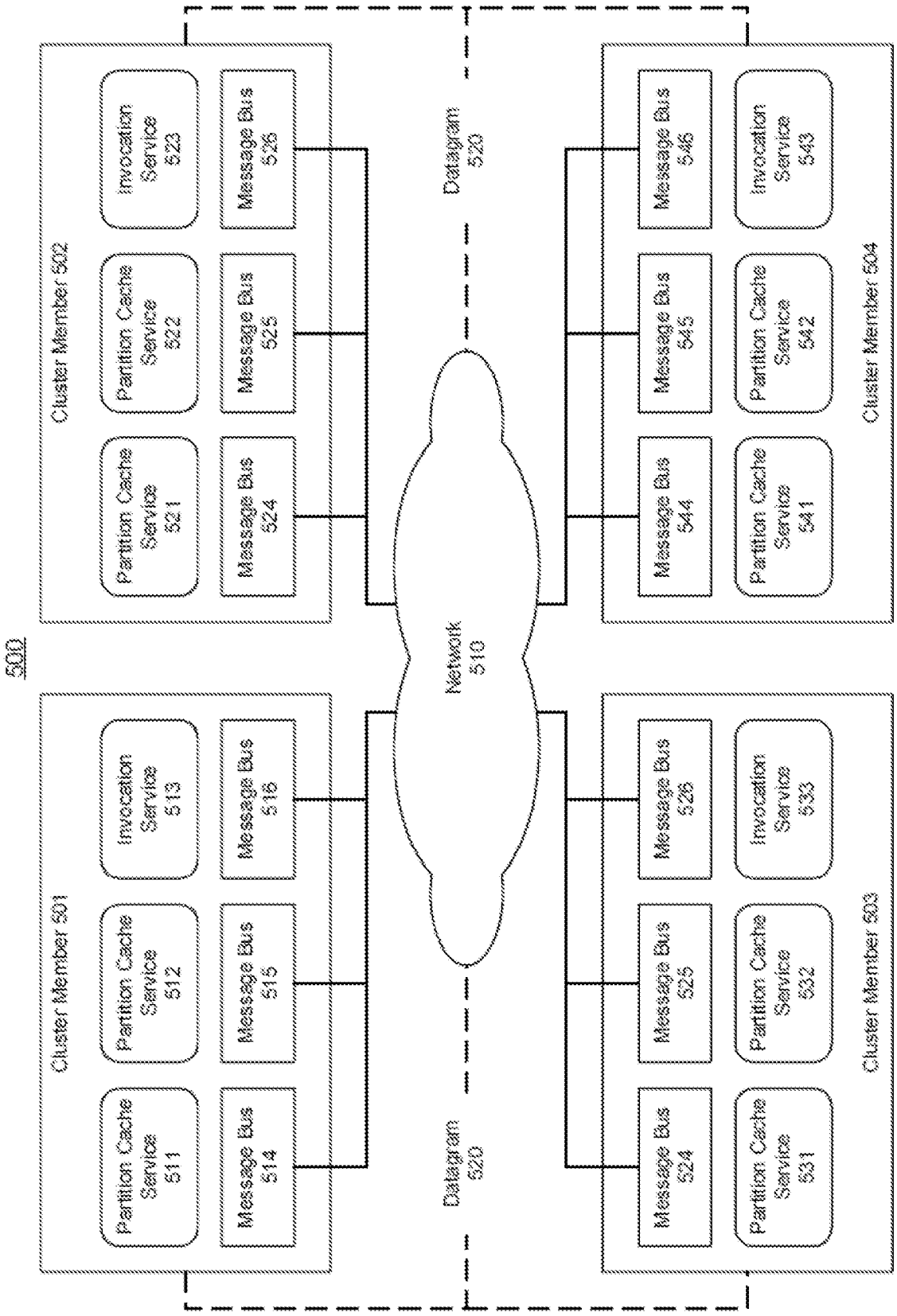
500
Network
510
Datagram
520
Datagram
520
Cluster Member 501
Partition Cache Service 511
Message Bus 514
Partition Cache Service 512
Message Bus 515
Invocation Service 513
Message Bus 516
Cluster Member 502
Partition Cache Service 521
Message Bus 524
Partition Cache Service 522
Message Bus 525
Invocation Service 523
Message Bus 526
Cluster Member 503
Partition Cache Service 531
Message Bus 524
Partition Cache Service 532
Message Bus 525
Invocation Service 533
Message Bus 526
Cluster Member 504
Partition Cache Service 541
Message Bus 544
Partition Cache Service 542
Message Bus 545
Invocation Service 543
Message Bus 546

FIGURE 5

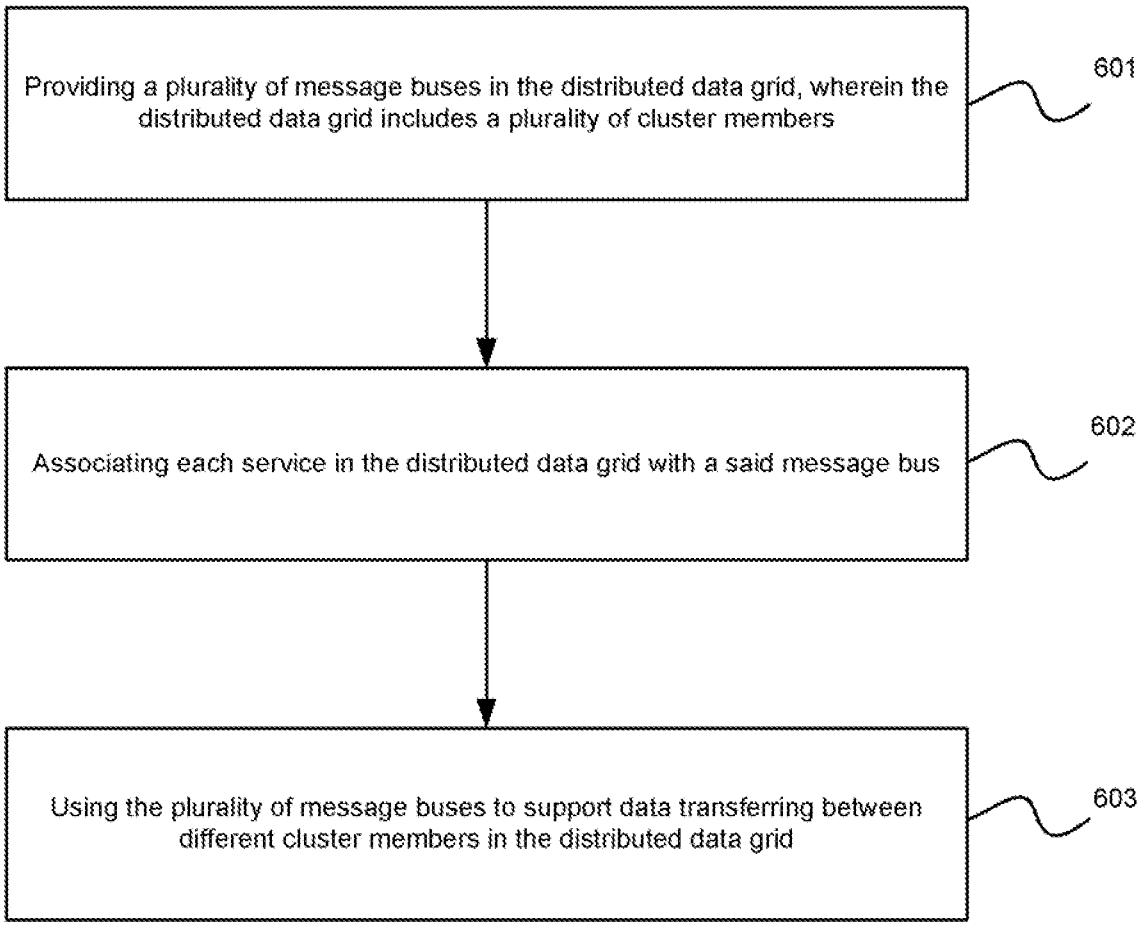
Providing a plurality of message buses in the distributed data grid, wherein the distributed data grid includes a plurality of cluster members
601
Associating each service in the distributed data grid with a said message bus
602
Using the plurality of message buses to support data transferring between different cluster members in the distributed data grid
603

FIGURE 6

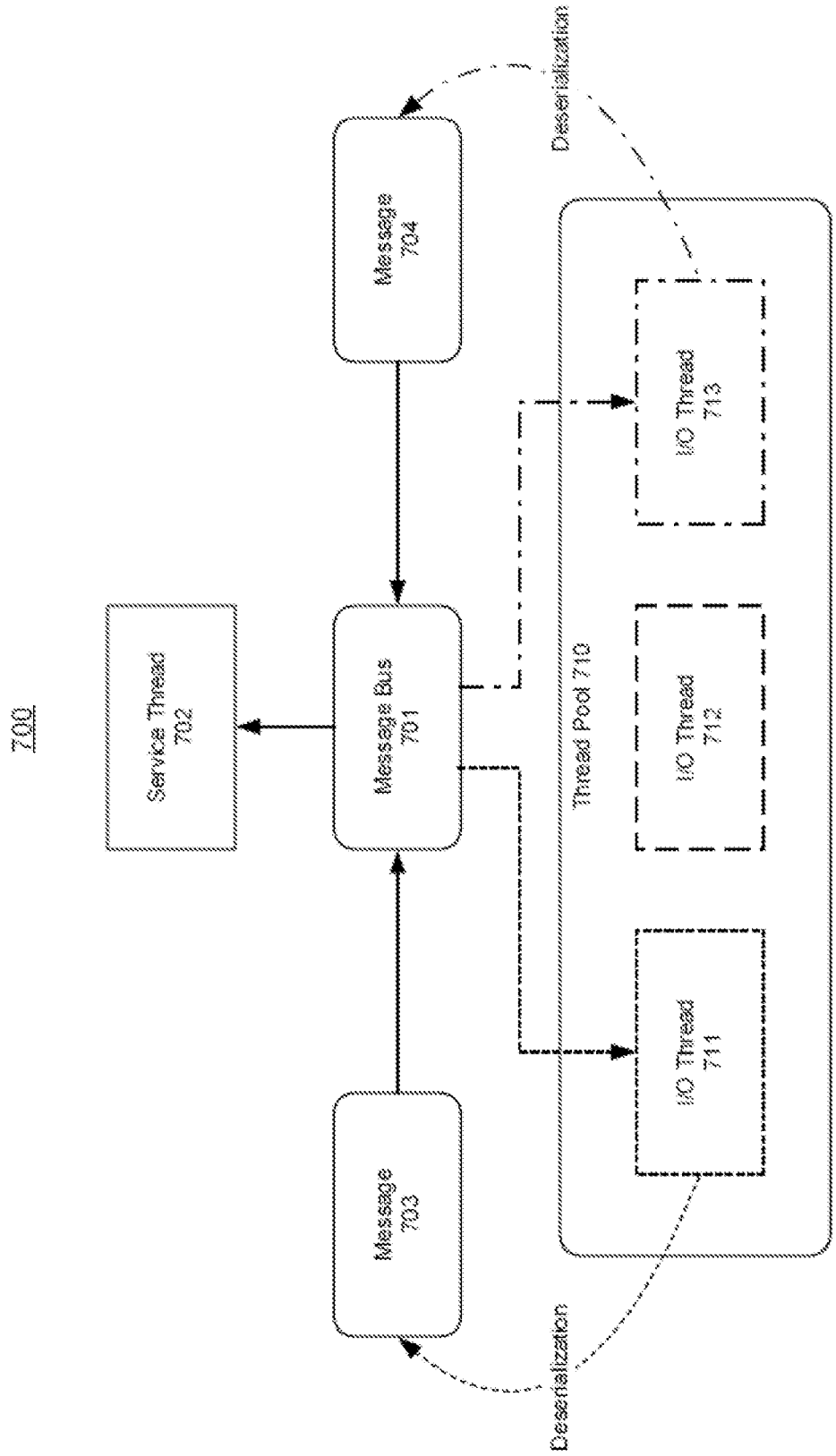
700
Service Thread
702
Message Bus
701
Message
704
Message
703
Thread Pool 710
I/O Thread
713
I/O Thread
712
I/O Thread
711
Deserialization
Deserialization

FIGURE 7

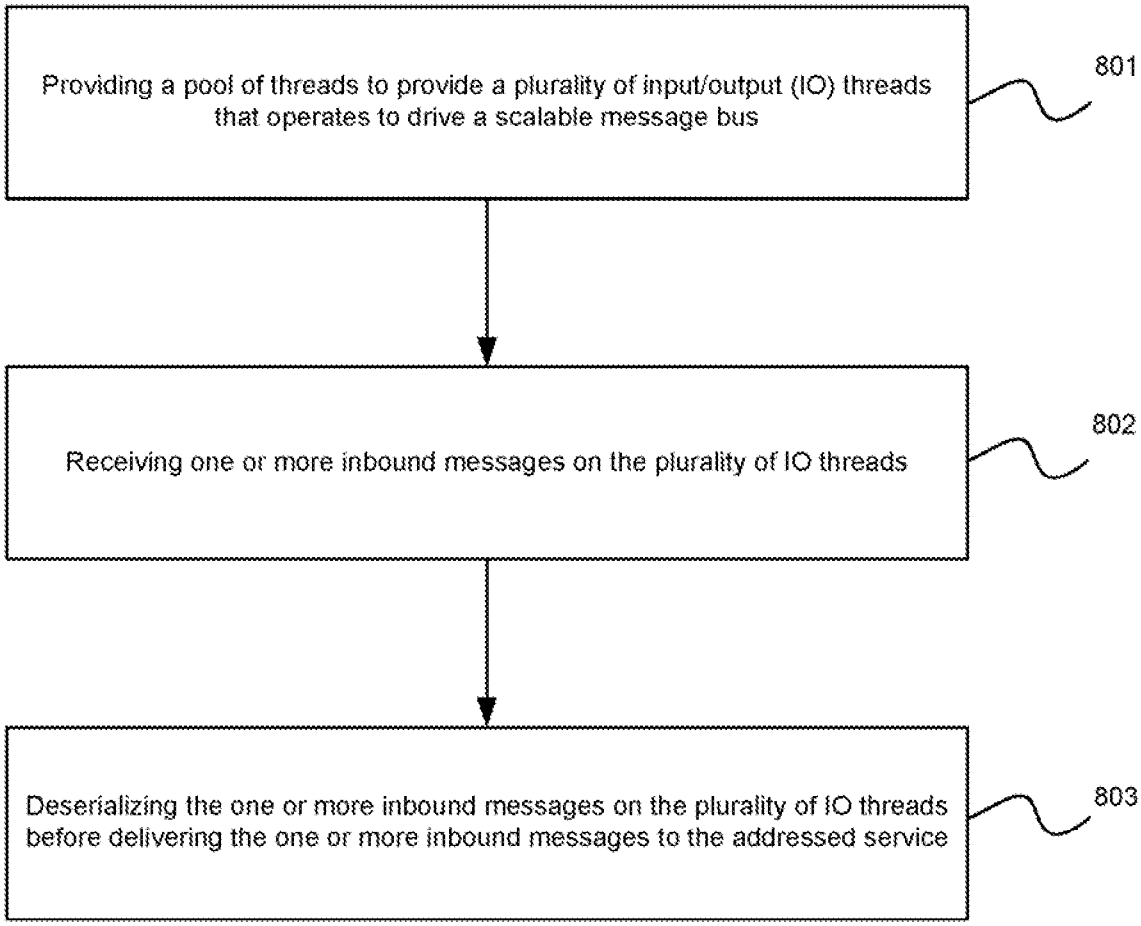
Providing a pool of threads to provide a plurality of input/output (IO) threads that operates to drive a scalable message bus
801
Receiving one or more inbound messages on the plurality of IO threads
802
Deserializing the one or more inbound messages on the plurality of IO threads before delivering the one or more inbound messages to the addressed service
803

FIGURE 8

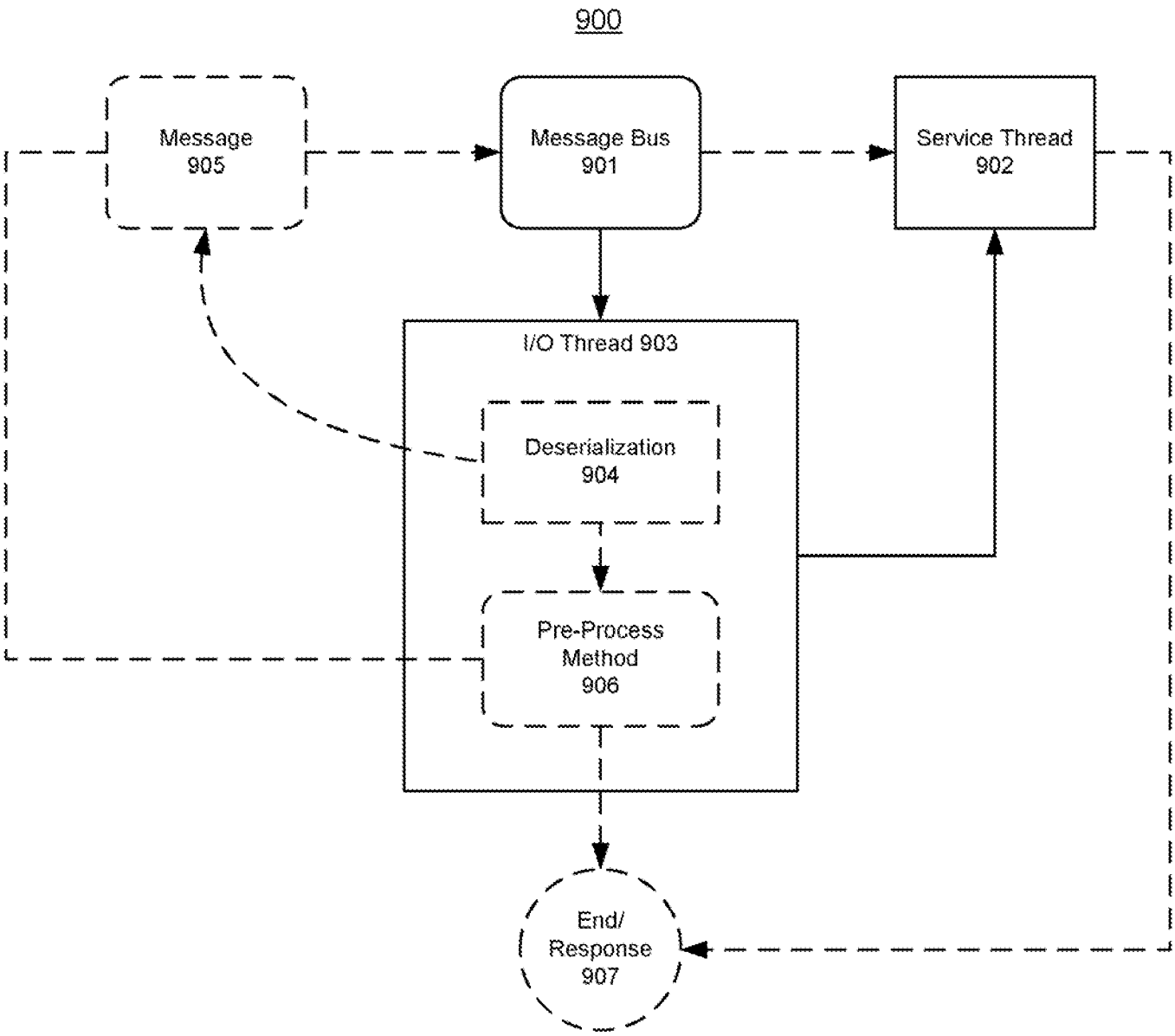
900
Message
905
Message Bus
901
Service Thread
902
I/O Thread 903
Deserialization
904
Pre-Process
Method
906
End/
Response
907

FIGURE 9

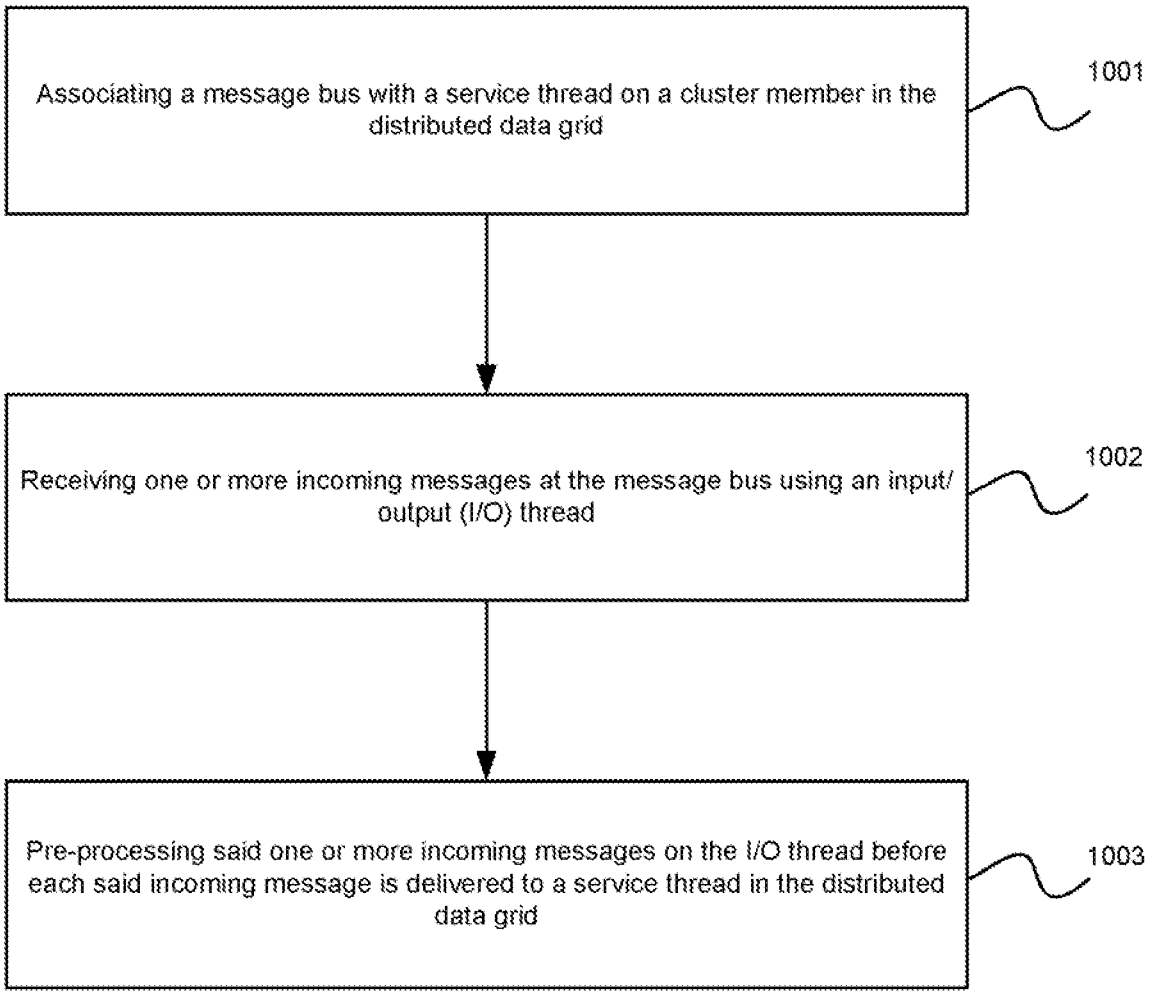
Associating a message bus with a service thread on a cluster member in the distributed data grid
1001
Receiving one or more incoming messages at the message bus using an input/output (I/O) thread
1002
Pre-processing said one or more incoming messages on the I/O thread before each said incoming message is delivered to a service thread in the distributed data grid
1003

FIGURE 10

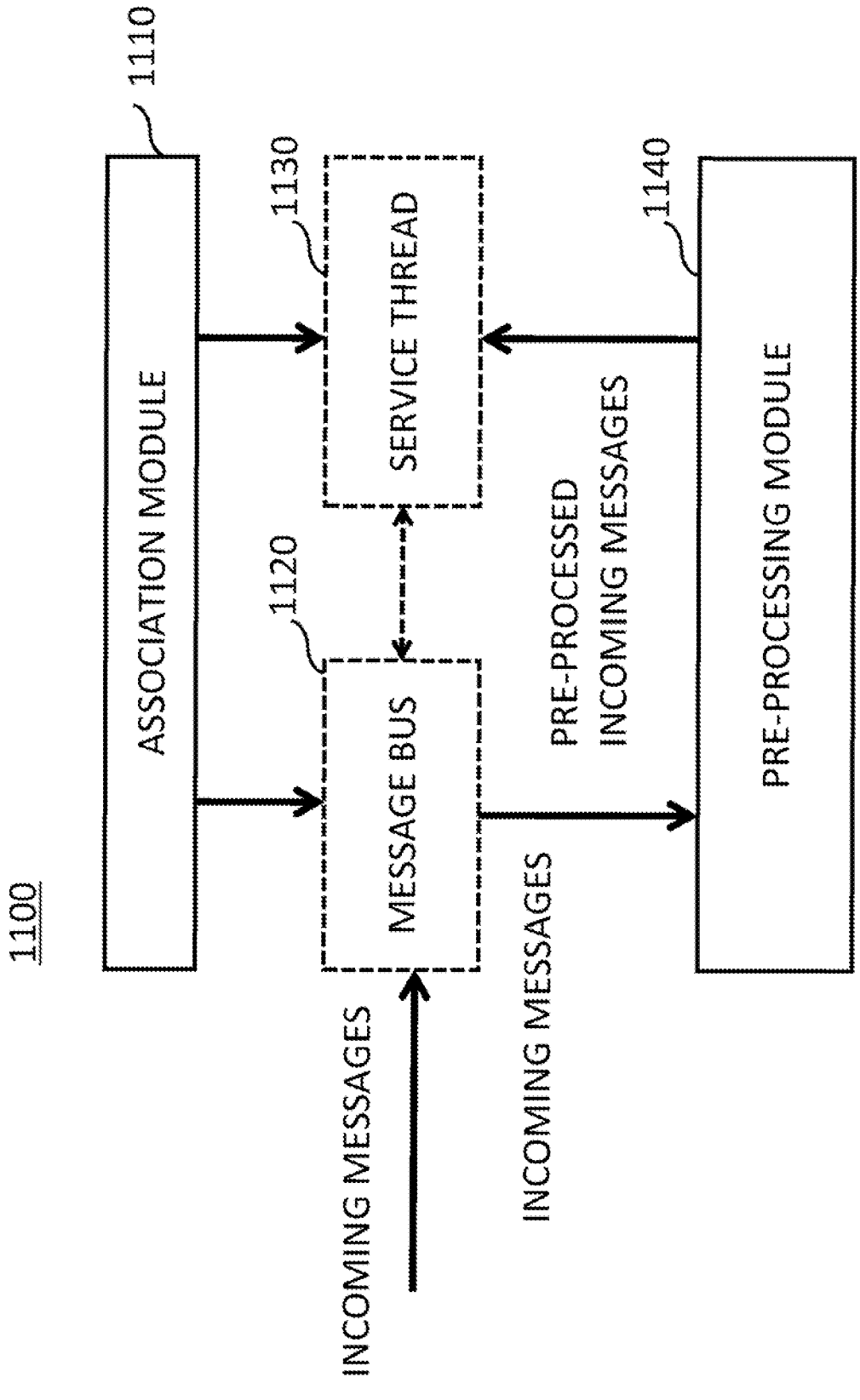
1100
1110
ASSOCIATION MODULE
1120
MESSAGE BUS
1130
SERVICE THREAD
1140
PRE-PROCESSED INCOMING MESSAGES
PRE-PROCESSING MODULE
INCOMING MESSAGES
INCOMING MESSAGES

FIGURE 11

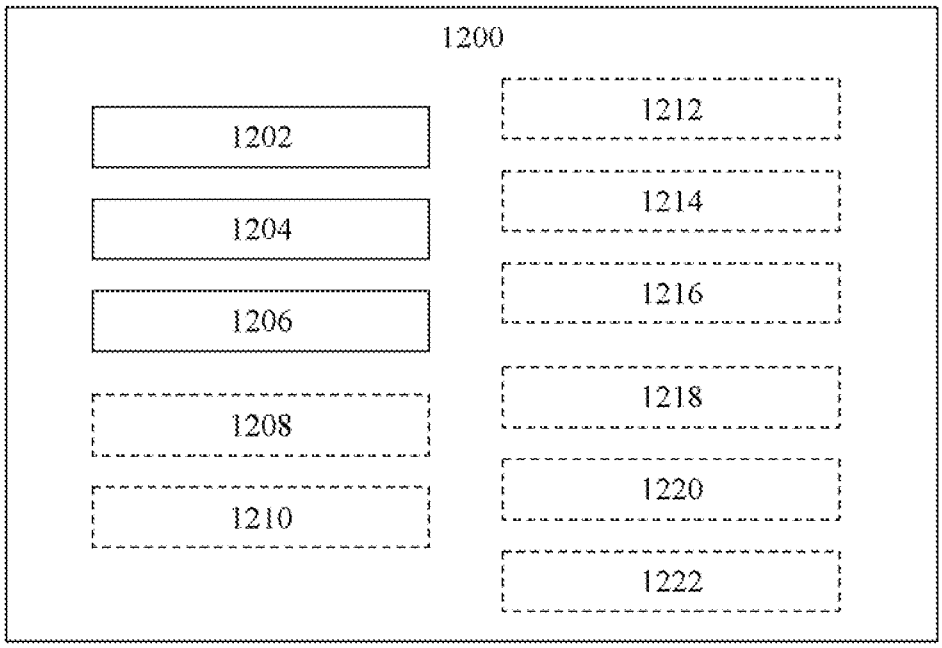

FIGURE 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2013/058610

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/08 G06F9/50 G06F9/455
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2005/080982 A1 (VASILEVSKY ALEXANDER D [US] ET AL) 14 April 2005 (2005-04-14) paragraph [0015] - paragraph [0018] paragraph [0025] paragraph [0077] - paragraph [0112] paragraph [0120] - paragraph [0138] paragraph [0147] - paragraph [0151] -----	1-23
A	US 2002/078126 A1 (HIGGINS THOMAS [IE]) 20 June 2002 (2002-06-20) paragraph [0009] - paragraph [0036] -----	1-23

-/--

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

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 December 2013	02/01/2014

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

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2013/058610

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	TABOADA G L ET AL: "Efficient Java Communication Libraries over InfiniBand", HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS, 2009. HPCC '09. 11TH IEEE INTERNATIONAL CONFERENCE ON, IEEE, PISCATAWAY, NJ, USA, 25 June 2009 (2009-06-25), pages 329-338, XP031491412, ISBN: 978-1-4244-4600-1 page 329, right-hand column, paragraph 2 - paragraph 3 page 330, left-hand column, paragraph 3 - right-hand column, paragraph 2 -----	1-23
A	SAYANTAN SUR ET AL: "RDMA read based rendezvous protocol for MPI over InfiniBand", PROCEEDINGS OF THE ELEVENTH ACM SIGPLAN SYMPOSIUM ON PRINCIPLES AND PRACTICE OF PARALLEL PROGRAMMING , PPOPP '06, 1 January 2006 (2006-01-01), page 32, XP055035735, New York, New York, USA DOI: 10.1145/1122971.1122978 ISBN: 978-1-59-593189-4 abstract page 32, left-hand column, last paragraph - page 33, left-hand column, paragraph 1 -----	1-23
A	EP 1 684 173 A1 (MICROSOFT CORP [US]) 26 July 2006 (2006-07-26) paragraph [0030] - paragraph [0036] paragraph [0040] - paragraph [0042] -----	1-23

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2013/058610

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005080982 A1	14-04-2005	US 2005080982 A1	14-04-2005
		WO 2006017584 A2	16-02-2006
US 2002078126 A1	20-06-2002	AU 5242600 A	02-01-2001
		EP 1186197 A1	13-03-2002
		GB 2355364 A	18-04-2001
		US 2002078126 A1	20-06-2002
		WO 0078089 A1	21-12-2000
EP 1684173 A1	26-07-2006	AU 2005246944 A1	20-07-2006
		BR PI0505537 A	12-09-2006
		CA 2531605 A1	30-06-2006
		CN 1798148 A	05-07-2006
		CO 5770119 A1	29-06-2007
		EP 1684173 A1	26-07-2006
		JP 2006190280 A	20-07-2006
		KR 20060079108 A	05-07-2006
		NZ 544320 A	28-09-2007
		SG 123734 A1	26-07-2006
		US 2006168269 A1	27-07-2006
		ZA 200510328 A	27-08-2008