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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 December 2015 (23.12.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/195079 A1

(51) International Patent Classification:
G06F 9/44 (2006.01) *G06F 13/14* (2006.01)

(21) International Application Number:
PCT/US2014/042563

(22) International Filing Date:
16 June 2014 (16.06.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventors: **LI, Jun**; 1501 Page Mill Rd., Palo Alto, California 94304 (US). **LAFFITTE, Hernan**; 1501 Page Mill Rd., Palo Alto, California 94304 (US). **BOLLINGER, Donald E**; 1501 Page Mill Rd., Palo Alto, California 94304 (US). **WU, Eric**; 1501 Page Mill Rd., Palo Alto, California 94304 (US).

(74) Agents: **BELL, Scott** et al.; Hewlett-Packard Company, Intellectual Property Administration, 3404 E. Harmony Road, Mail Stop 35, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, 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).

Declarations under Rule 4.17:

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

[Continued on next page]

(54) Title: VIRTUAL NODE DEPLOYMENTS OF CLUSTER-BASED APPLICATIONS

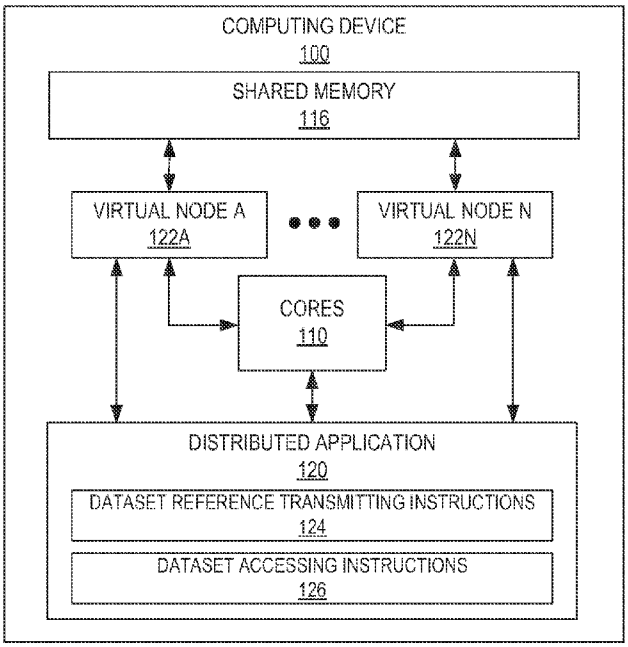

FIG. 1

(57) Abstract: Examples relate to deploying distributed applications using virtual nodes. In some examples, virtual nodes are created and are each assigned a core subset of a number of processing cores, an Internet protocol (IP) address, and an in-memory file system configured to provide access to a portion of physically shared memory. At this stage, a distributed application that is configured to be deployed to a plurality of machine nodes is deployed to the plurality of virtual nodes. On a first virtual node, a reference to a first dataset stored in physically shared memory is sent to a second virtual node, where the physically shared memory is accessible to each of the plurality of virtual nodes. Next, on the second virtual node, the first dataset is accessed through the in-memory file system of the first virtual node.

WO 2015/195079 A1

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

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

VIRTUAL NODE DEPLOYMENTS OF CLUSTER-BASED APPLICATIONS

Background

[0001] In-memory and multicore computing is becoming more prevalent in today's information technology (IT) industry. Customers can be offered with more powerful computers equipped with larger amount of memory to provide IT applications (e.g., real-time data analytics). Immediate performance gains can be achieved by deploying cluster-based applications (i.e., distributed applications) to a single machine environment that includes one OS image, a large number of processor cores, and a large amount of memory. Typically, the cluster-based application is migrated or rewritten to be compatible with the single machine environment, which involves, for example, source code changes, deployment scripts, etc.

Brief Description of the Drawings

[0002] The following detailed description references the drawings, wherein:

[0003] FIG. 1 is a block diagram of an example computing device for providing a cluster-based application on virtual nodes in a single machine environment;

[0004] FIG. 2 is a block diagram of an example computing device that includes an application-management application for providing virtual node deployments of cluster-based applications;

[0005] FIG. 3 is a flowchart of an example method for execution by a computing device for providing virtual node deployments of cluster-based applications; and

[0006] FIG. 4 is a workflow diagram showing an example virtual node deployment that is configured to provide a cluster-based application.

Detailed Description

[0007] As discussed above, cluster-based applications can be migrated to single machine environments. For example, a cluster-based application can be completely re-designed to take advantage of shared-memory and inter-thread

communication (ITC) in a single machine environment. In another example, the cluster-based application can be installed on a typical machine cluster and modified to support virtual shared memory that exists across the machines in the cluster.

[0008] Single machine environments provide operating system-level virtualization and/or other lower-level containers that allow for multiple execution environments to simultaneously exist in the single machine environment. For example, a container may be a light-weight virtualized environment managed by a single OS image, which better supports resource allocation and resource isolation. In this example, control groups supported by the OS may allocate resources such as CPU, system memory, block IO devices and network bandwidth to user processes based on sharing policies.

[0009] In examples described herein, a virtual node solution using OS containers is provided, where each formerly distributed machine node is implemented as a virtual node. Container features are used to manage allocation of in-memory file systems, which are localized to optimize memory access for each node. Further, per-container IP address assignments are used in the virtual nodes to provide fast inter-process communication (IPC) between virtual nodes using the same interface the cluster-based application used in its original distributed environment. The virtual node deployment provides a general solution for the deployment of cluster-based applications in single machine environments such as a large multicore big-memory system.

[0010] In some examples, virtual nodes are created and are each assigned a core subset of a number of processing cores, an Internet protocol (IP) address, and an in-memory file system configured to provide access to a portion of physically shared memory, where the physically shared memory is accessible to each of the plurality of virtual nodes. At this stage, a cluster-based application that is configured to be deployed to a plurality of machine nodes is deployed to the plurality of virtual nodes. On a first virtual node, a reference to a first dataset stored in the physically shared memory is sent to a second virtual node.

Next, on the second virtual node, the first dataset is accessed through the in-memory file system of the first virtual node.

[0011] Referring now to the drawings, FIG. 1 is a block diagram of an example computing device 100 for providing a cluster-based application on virtual nodes in a single machine environment. The computing device 100 provides a single machine environment and may be implemented as, for example, a large multicore big-memory system. In the example of FIG. 1, computing device 100 includes cores 110, shared memory 116, distributed application 120, and virtual nodes 122A, 122N.

[0012] Cores 110 may be any number of central processing units (CPUs), microprocessors, processing cores, and/or other hardware devices suitable for retrieval and execution of instructions stored in a machine-readable storage medium. Cores 110 may fetch, decode, and execute instructions 124, 126 to enable the distributed application 120 to be executed on virtual nodes 122A, 122N, as described below. As an alternative or in addition to retrieving and executing instructions, cores 110 may include any number of electronic circuits including a number of electronic components for performing the functionality of instructions 124 and/or 126.

[0013] Shared memory 116 is physically shared memory that is accessible to virtual nodes 122A, 122N. Shared memory 116 may be random access memory (RAM) that provides, for example, non-uniform memory access. Shared memory 116 may be encoded with executable instructions from distributed application 120.

[0014] Distributed application 120 is a cluster-based application that is configured to be installed on a cluster of machine nodes. Examples of distributed applications 120 include HADOOP®, High Performance Computing Cluster, etc. HADOOP® is a framework for storing and processing large-scale datasets (e.g., file-based data records, database records, etc.) on a cluster. HADOOP® is a registered trademark of Apache Software Foundation, which is headquartered in Forest Hill, Maryland.

[0015] Virtual nodes 122A, 122N are virtualized servers that are provided by operating system containers. For example, each virtual node (e.g., virtual node A

122A, virtual node N 122N) may be a light-weight virtualized environment managed by a LINUX® OS instance that supports resource allocation and resource isolation. Specifically, each virtual node (e.g., virtual node A 122A, virtual node N 122N) can be assigned a subset of cores 110, an internet protocol (IP) address, and an in-memory file system for accessing a portion of shared memory 116. The resources can be assigned when the virtual nodes 122A, 122N are initially configured to provide distributed application 120. LINUX® is a registered trademark of Linus Torvalds, an individual living in Portland, Oregon.

[0016] During the execution of distributed application 120, virtual nodes 122A, 122N may transmit dataset references to each other to allow multiple virtual nodes 122A, 122N to access a common dataset in shared memory 116. Dataset reference transmitting instructions 124 transmit a dataset reference from, for example, virtual node A 122A to virtual node N 122N. The dataset reference may include a location to access the common dataset stored in shared memory. For example, the dataset reference may include a file path for accessing the common dataset and an identifier for an in-memory file system that has access to a portion of the shared memory where the common dataset is stored.

[0017] Dataset accessing instructions 126 may be executed by virtual node N 122N to access the common dataset. The dataset reference received from virtual node A 122A may be used to identify the in-memory file system, which is then used to access the common dataset. The in-memory file system may be assigned to virtual node A 122A to minimize memory access time of datasets that are frequently accessed by virtual node A 122A.

[0018] FIG. 2 is a block diagram of an example computing device 200 including shared memory 216, distributed application 220, virtual nodes 222A, 222N, and application-management application 230. The components of computing device 200 may be similar to the corresponding components of computing device 100 described with respect to FIG. 1.

[0019] Application-management application 230 configures the virtual nodes 222A, 222N of computing device 200. Specifically, application-management application 230 creates virtual nodes 222A, 222N and assigns resources of

computing device 200 to each virtual node. In this example, virtual node A 222A has been assigned cores 210A, 210B, IP address A 226A, and in-memory file system A 224A, which provides virtual node A 222A with access to portion A 218A of shared memory 216. Similarly, virtual node N 222N has been assigned cores 210C, 210D, IP address N 226N, and in-memory file system N 224N, which provides virtual node N 222N with access to portion N 218N of shared memory 216. Each virtual node 222A, 222N has access to the in-memory file systems 224A, 224N of the other virtual node. The access to the in-memory file systems 224A, 224N may be provided by application-management application 230, which acts as the primary operating system of computing device 200 and manages execution environments for the virtual nodes 222A, 222N. Further, each virtual node 222A, 222N can be assigned any number of process nodes (not shown) of distributed application 220, where each process node provides a service (e.g., HADOOP® distributed file system (HDFS™) or MapReduce in the case of HADOOP®, etc.) of distributed application. In FIG. 2, two virtual nodes 222A, 222N are shown, but computing device 200 can include any number of virtual nodes. HDFS™ is a registered trademark of Apache Software Foundation, which is headquartered in Forest Hill, Maryland.

[0020] Shared memory 216 may be non-uniform memory where memory access time is dependent on a location in the shared memory 216 relative to the accessing core (e.g., core A 210A, core B 210B, core C 210C, core D 210D). In this case, each portion (e.g., portion A 218A, portion N 218N, etc.) of shared memory 216 is assigned to a virtual node (e.g., virtual node A 222A, virtual node N 222N) such that memory access times are minimized. Further, the use of multiple in-memory file systems 224A, 224N reduces locking contention when, for example, attempting to update the metadata of data within shared memory 216 if multiple readers and writers are launched from multiple virtual nodes simultaneously (e.g., virtual node A 222A, virtual node N 222N). For example, a Linux OS command can be used to create temporary file storage facilities ("tmpfs") as in-memory file systems 224A, 224N with the command option of "-o mpol=bind:<processor id>" to explicitly bind the in-memory file system 224A, 224N to a processor.

[0021] IP addresses 226A, 226N allows virtual nodes 222A, 222N to communicate with each other via typical TCP/IP-based channels used by distributed application 220. In other cases, other types of addresses may be used such as addresses that are compliant with a corresponding inter-process communication protocol. In a typical cluster environment, a process node may be bound to the machine node's IP address. In a single machine environment like computing device 200, multiple IP addresses 226A, 226N are created such that one IP address can be assigned to each virtual node (e.g., virtual node A 222A, virtual node N 222N). A process node bound to the virtual node (e.g., virtual node A 222A, virtual node N 222N) can then use the assigned IP address for data communication. For example, IP address aliasing can be used to create multiple IP addresses 226A, 226N on computing device 200. In some cases, multiple IP addresses 226A, 226N are created from a physical network card. In other cases, when a machine has multiple physical network cards, different IP address groups can be created and bound to a different network card, where a virtual node (e.g., virtual node A 222A, virtual node N 222N) is assigned an IP address 226A, 226N from the IP address groups.

[0022] Cores 210A-210D are cores of multi-core processors configured for single machine environments. Computing device 200 includes multiple processors (i.e., sockets), and each processor has multiple processing cores. Any number of cores may be assigned to each of the virtual nodes 222A, 222N. For example, a virtual node (e.g., virtual node A 222A, virtual node N 222N) can be assigned with all of the cores (e.g., core A 210A – core B 210B, core C 210C – core D 210D) that belong to a particular socket. Alternatively, a virtual node (e.g., virtual node A 222A, virtual node N 222N) can be assigned with only a portion of the cores (e.g., core A 210A, core C 210C) that belong to the particular socket. Application-management application 230 may be configured to ensure that a virtual node (e.g., virtual node A 222A, virtual node N 222N) is restricted to the cores allocated to the virtual node. In the case of a Linux OS, a "numactl -membind <socket id> -physcpubind <the cpu cores>" command can be used to confine a virtual node (e.g., virtual node A 222A, virtual node N 222N) to run on the specified cores. The "-membind" option specifies that all process nodes

executing on the virtual node (e.g., virtual node A 222A, virtual node N 222N) should access a portion (e.g., portion A 218A, portion N 218N, etc.) of shared memory 216 bound to the specified processor. Similarly, the option of "-physcpubind" specifies the cores (e.g., core A 210A, core B 210B, core C 210C, core D 210D) that are bound to the virtual node (e.g., virtual node A 222A, virtual node N 222N).

[0023] Application-management application 230 is configured to deploy distributed application 220 to virtual nodes 222A, 222N. Distributed application 220 is a cluster-based application that is configured to be installed on a cluster of machine nodes and, for example, provide a framework for storing and processing large-scale datasets. In this case, a machine node is a physical machine with a distinct operating environment and hardware (e.g., a server, a desktop computer, etc.). In FIG. 2, distributed application 220 may be adapted to be deployed on virtual nodes 222A, 222N rather than physical machine nodes. For example, because virtual nodes 222A, 222N have physically shared memory, distributed application 220 may be adapted to pass large data sets between virtual nodes 222A, 222N using references rather than actually transferring the large data sets. In this example, performance of the distributed application is improved by reducing the latency of inter-node communications. In the case that distributed application 220 is HADOOP®, user-defined applications created using, for example, MapReduce can be deployed in the HADOOP® environment provided by distributed application 220 without modification.

[0024] Distributed application 220 executes on the virtual nodes 222A, 222N to provide access to large-scale datasets. For example, a virtual node A 222A may initiate a process for creating or modifying a dataset (e.g., file-based data records, database records, etc.), which is performed in a portion A 218A of shared memory 216. In this example, the process launched from virtual node A 222A may be performed on a subset of cores (e.g., core A 210A, core B 210B) of virtual node A 222A and may use in-memory file system A 224 to access portion A 218A. Virtual node A 222A may be configured to provide other virtual nodes (e.g., virtual node N 222N) on computing device 200 with access to the dataset by sending the other virtual nodes a reference to the dataset. The reference may identify in-memory

file system A 224A and include a location (e.g., memory address, file path, etc.) of the dataset in portion A 218A, where the other virtual nodes (e.g., virtual node N 222N) can use the reference to access the dataset directly in shared memory 216 without transmitting a portion of or the entire dataset.

[0025] FIG. 3 is a flowchart of an example method 300 for execution by a computing device 100 for providing virtual node deployments of cluster-based applications. Although execution of method 300 is described below with reference to computing device 100 of FIG. 1, other suitable devices for execution of method 300 may be used, such as computing device 200 of FIG. 2. Method 300 may be implemented in the form of executable instructions stored on a machine-readable storage medium and/or in the form of electronic circuitry.

[0026] Method 300 may start in block 305 and continue to block 310, where computing device 100 creates virtual nodes and assigns resources to the virtual nodes. Specifically, each virtual node may be assigned cores, an IP address, and in-memory file system. In block 315, a distributed application that is configured for a cluster of machine nodes is deployed to the virtual nodes. In some cases, the distributed application is modified to optimize performance for a single machine environment before it is deployed to the virtual nodes.

[0027] In block 320, a first virtual node of computing device 100 sends a dataset reference to a second virtual node. The dataset reference may include a location in shared memory of the common dataset to be shared. In block 325, a second virtual node of computing device accesses the common dataset through an in-memory file system that is assigned to the first virtual node. Method 300 may then proceed to block 330, where method 300 ends.

[0028] FIG. 4 is a workflow diagram 400 showing an example virtual node deployment that is configured to provide a cluster-based application. Workflow diagram 400 shows shared memory 402, virtual node A 404A, virtual node N 404N, and application-management application 406, which may each be similar to their corresponding components described above with respect to FIGS. 1 and 2.

[0029] In block 420, application-management application 406 creates virtual node A 404A and then assigns resources (e.g., IP address, processing cores, in-memory file system, etc.) to virtual node A 404A. In block 422, application-management application 406 creates virtual node N 404N and then assigns resources (e.g., IP address, processing cores, in-memory file system, etc.) to virtual node N 404N.

[0030] In block 424, application-management application 406 deploys a distributed application to virtual node A 404A and virtual node N 404N. The distributed application was originally configured to be installed on a cluster of machine nodes, and in this example, each of the virtual nodes 404A, 404N acts as a machine node so that the distributed application can be installed in a single machine environment such as computing device 200 of FIG. 2.

[0031] In block 426, virtual node A 404A accesses a first in-memory file system that manages data in a first portion of shared memory 402. The first in-memory file system may be mounted and shared to the virtual nodes 404A, 404N by application-management application 406 when resources were assigned in block 420. In block 428, virtual node A 404A uses the first in-memory files system to create a first file in shared memory 402. Virtual node A 404A may then proceed to modify and close the first file in shared memory 404 in blocks 430 and 432.

[0032] In block 434, virtual node A 404A sends a first file reference to virtual node N 404N. In block 436, virtual node N 404N accesses the first in-memory file system that manages data in the first portion of shared memory 402. In block 438, virtual node N 404N uses the first in-memory files system to access the first file in shared memory 402. Virtual node N 404N may then proceed to read and close the first file in shared memory 404 in blocks 440 and 442.

[0033] In block 444, virtual node N 404N accesses a second in-memory file system that manages data in a second portion of shared memory 402. Similar to the first in-memory file system, the second in-memory file system may be mounted and shared to the virtual nodes 404A, 404N by application-management application 406 when resources were assigned in block 422. In block 446, virtual

node N 404N uses the second in-memory files system to create a second file in shared memory 402.

[0034] The foregoing disclosure describes a number of examples for deploying cluster-based applications using virtual nodes. In this manner, the examples disclosed herein facilitate deploying a distributed application to a single machine environment by using virtual nodes that are assigned resources to simulate a cluster of machine nodes.

CLAIMS

We claim:

1. A system for deploying cluster-based applications using virtual nodes, comprising:

a plurality of operating system containers that each provides an isolated environment for one of a plurality of virtual nodes;

physically shared memory that is accessible to each of the plurality of virtual nodes;

an application-management application that is executed to:

create the plurality of virtual nodes that are each assigned a core subset of a plurality of processing cores, an Internet protocol (IP) address of a plurality of IP addresses, and an in-memory file system of a plurality of in-memory file systems configured to provide access to a portion of the physically shared memory; and

deploy a distributed application that is configured to be deployed to a plurality of machine nodes to the plurality of virtual nodes; and

the distributed application that is executed to:

on a first virtual node of the plurality of virtual nodes, send a reference to a first dataset stored in the physically shared memory to a second virtual node of the plurality of virtual nodes; and

on the second virtual node, access the first dataset through the in-memory file system of the first virtual node.

2. The system of claim 1, wherein the physically shared memory is non-uniform memory, and wherein the core subset is assigned to the first virtual node to minimize a memory access time that is based on a relative location of the core subset of the first virtual node with respect to the portion of the non-uniform memory assigned to the first virtual node.

3. The system of claim 1, wherein the distributed application is further executed to:

on the second virtual node, receive a request to create a second dataset in the physically shared memory; and

on the second virtual node, use the in-memory file system of the second virtual node to create the second dataset in the portion of the non-uniform memory assigned to the second virtual node.

4. The system of claim 1, wherein each of the plurality of virtual nodes has access to the in-memory file system of each of the plurality of virtual nodes.

5. The system of claim 1, wherein the reference to the first dataset identifies the in-memory file system of the first virtual node.

6. The system of claim 1, wherein the distributed application is modified to transmit the reference to the first dataset rather than the first dataset.

7. A method for deploying cluster-based applications using virtual nodes, comprising:

creating a plurality of virtual nodes that are each assigned a core subset of a plurality of processing cores, an Internet protocol (IP) address of a plurality of IP addresses, and an in-memory file system of a plurality of in-memory file systems configured to provide access to a portion of physically shared memory, wherein the physically shared memory is accessible to each of the plurality of virtual nodes;

deploying a distributed application that is configured to be deployed to a plurality of machine nodes to the plurality of virtual nodes;

on a first virtual node of the plurality of virtual nodes, sending a reference to a first dataset stored in the physically shared memory to a second virtual node of the plurality of virtual nodes;

on the second virtual node, accessing the first dataset through the in-memory file system of the first virtual node; and

on the second virtual node, using the in-memory file system of the second virtual node to create a second dataset in the portion of the non-uniform memory assigned to the second virtual node

8. The method of claim 7, wherein the physically shared memory is non-uniform memory, and wherein the core subset is assigned to the first virtual node to minimize a memory access time that is based on a relative location of the core subset of the first virtual node with respect to the portion of the non-uniform memory assigned to the first virtual node.

9. The method of claim 7, wherein each of the plurality of virtual nodes has access to the in-memory file system of each of the plurality of virtual nodes.

10. The method of claim 7, wherein the reference to the first dataset identifies the in-memory file system of the first virtual node.

11. The method of claim 7, wherein the distributed application is modified to transmit the reference to the first dataset rather than the first dataset.

12. A non-transitory machine-readable storage medium encoded with instructions executable by a processor for deploying cluster-based applications using virtual nodes, the machine-readable storage medium comprising instructions to:

create a plurality of virtual nodes that are each assigned a core subset of a plurality of processing cores, an Internet protocol (IP) address of a plurality of IP addresses, and an in-memory file system of a plurality of in-memory file systems configured to provide access to a portion of physically shared memory, wherein the physically shared memory is accessible to each of the plurality of virtual nodes;

deploy a distributed application that is configured to be deployed to a plurality of machine nodes to the plurality of virtual nodes, wherein each of the

plurality of virtual nodes has access to the in-memory file system of each of the plurality of virtual nodes;

on a first virtual node of the plurality of virtual nodes, send a reference to a first dataset stored in the physically shared memory to a second virtual node of the plurality of virtual nodes;

on the second virtual node, access the first dataset through the in-memory file system of the first virtual node; and

on the second virtual node, use the in-memory file system of the second virtual node to create a second dataset in the portion of the non-uniform memory assigned to the second virtual node

13. The non-transitory machine-readable storage medium of claim 12, wherein the physically shared memory is non-uniform memory, and wherein the core subset is assigned to the first virtual node to minimize a memory access time that is based on a relative location of the core subset of the first virtual node with respect to the portion of the non-uniform memory assigned to the first virtual node.

14. The non-transitory machine-readable storage medium of claim 12, wherein the reference to the first dataset identifies the in-memory file system of the first virtual node.

15. The non-transitory machine-readable storage medium of claim 12, wherein the distributed application is modified to transmit the reference to the first dataset rather than the first dataset.

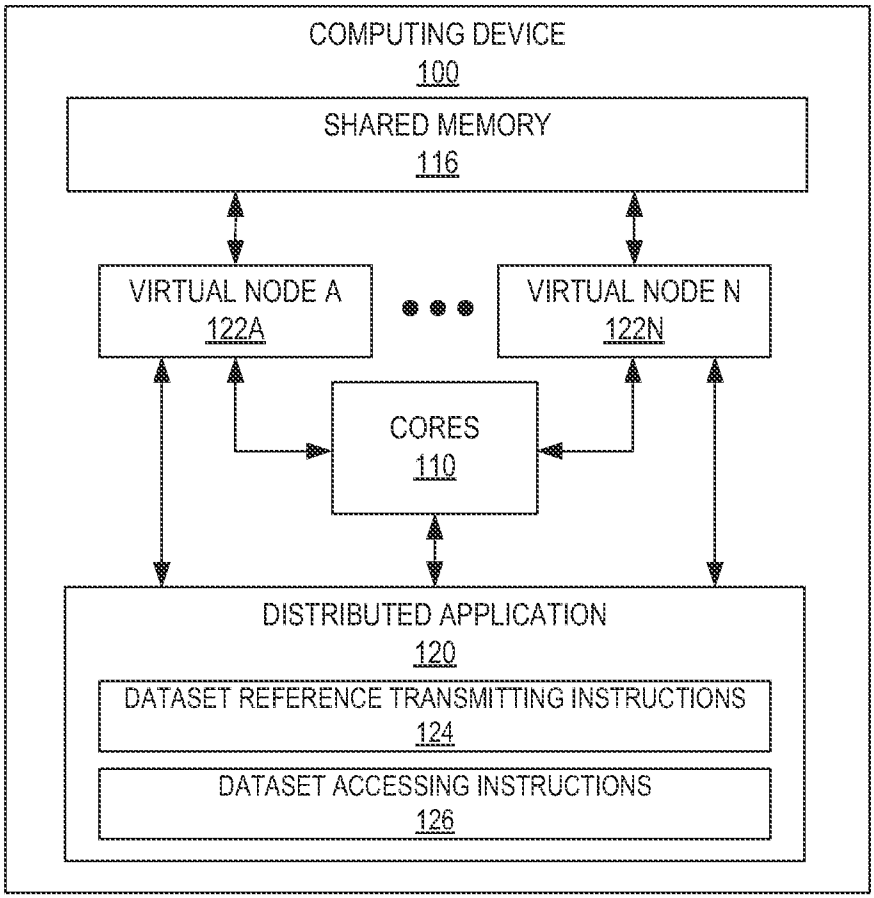
COMPUTING DEVICE
100
SHARED MEMORY
116
VIRTUAL NODE A
122A
VIRTUAL NODE N
122N
CORES
110
DISTRIBUTED APPLICATION
120
DATASET REFERENCE TRANSMITTING INSTRUCTIONS
124
DATASET ACCESSING INSTRUCTIONS
126

FIG. 1

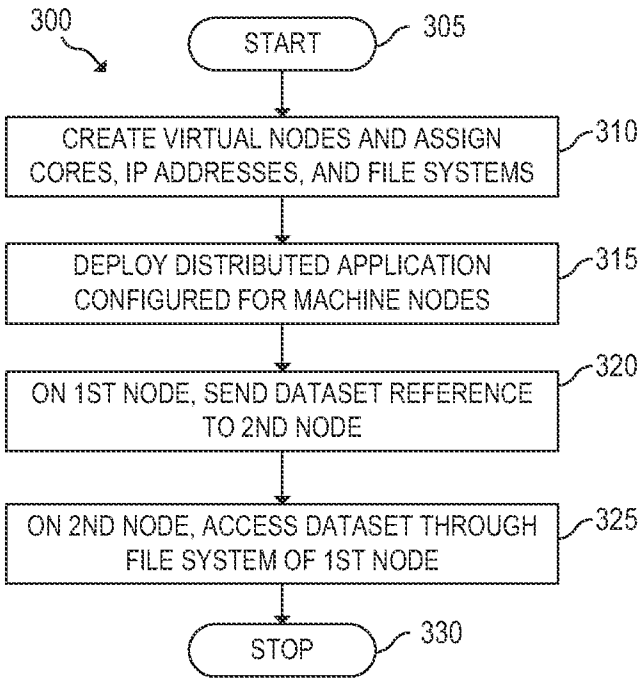
300
START
305
CREATE VIRTUAL NODES AND ASSIGN CORES, IP ADDRESSES, AND FILE SYSTEMS
310
DEPLOY DISTRIBUTED APPLICATION CONFIGURED FOR MACHINE NODES
315
ON 1ST NODE, SEND DATASET REFERENCE TO 2ND NODE
320
ON 2ND NODE, ACCESS DATASET THROUGH FILE SYSTEM OF 1ST NODE
325
STOP
330

FIG. 3

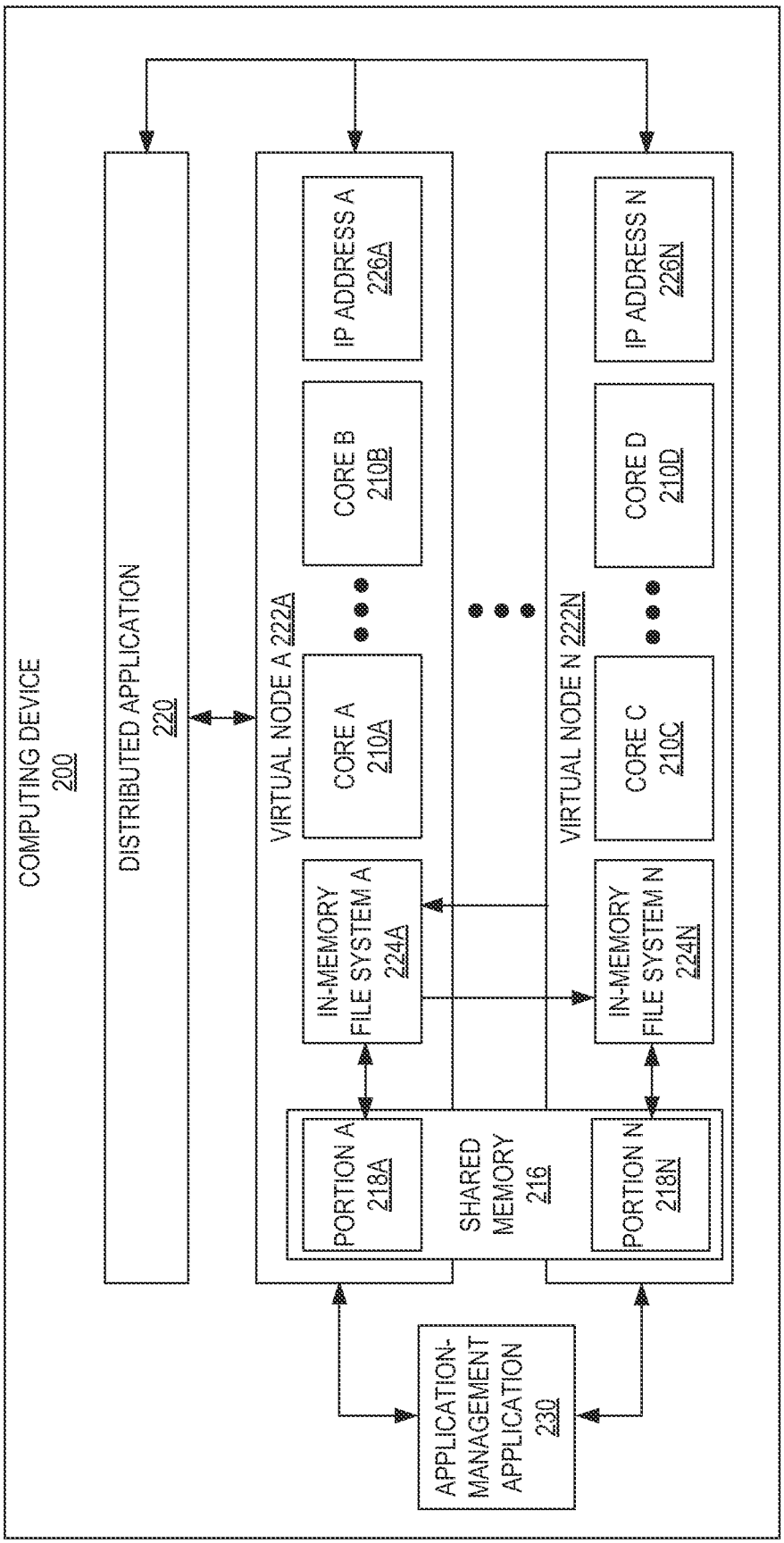
COMPUTING DEVICE
200
DISTRIBUTED APPLICATION
220
VIRTUAL NODE A 222A
IP ADDRESS A
226A
CORE B
210B
CORE A
210A
IN-MEMORY
FILE SYSTEM A
224A
VIRTUAL NODE N 222N
IP ADDRESS N
226N
CORE D
210D
CORE C
210C
IN-MEMORY
FILE SYSTEM N
224N
PORTION A
218A
SHARED
MEMORY
216
PORTION N
218N
APPLICATION-
MANAGEMENT
APPLICATION
230

FIG. 2

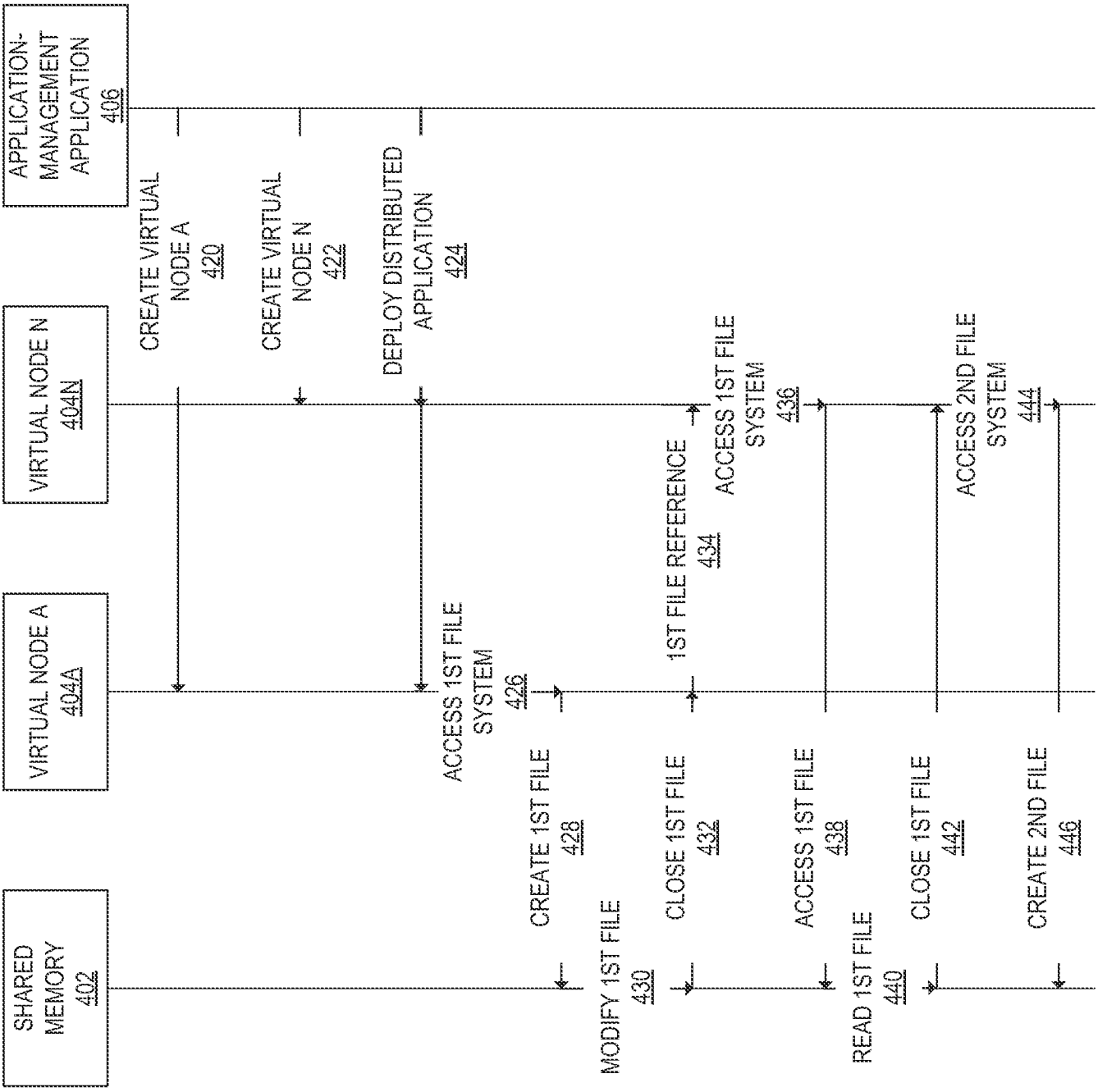
APPLICATION-MANAGEMENT APPLICATION 406
VIRTUAL NODE N 404N
VIRTUAL NODE A 404A
SHARED MEMORY 402
CREATE VIRTUAL NODE A 420
CREATE VIRTUAL NODE N 422
DEPLOY DISTRIBUTED APPLICATION 424
ACCESS 1ST FILE SYSTEM 426
CREATE 1ST FILE 428
MODIFY 1ST FILE 430
CLOSE 1ST FILE 432
1ST FILE REFERENCE 434
ACCESS 1ST FILE SYSTEM 436
ACCESS 1ST FILE 438
READ 1ST FILE 440
CLOSE 1ST FILE 442
ACCESS 2ND FILE SYSTEM 444
CREATE 2ND FILE 446

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/042563

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/44(2006.01)i, G06F 13/14(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 9/44; G06F 12/10; G06F 12/00; G06F 9/455; G06F 12/08; G06F 13/14

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: virual, node, distributed, application

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2005-0273571 A1 (THOMAS LYON et al.) 08 December 2005 See abstract; paragraphs [0022]-[0039]; and figures 2-5.	1-15
A	US 2005-0039180 A1 (SHAI FULTHEIM et al.) 17 February 2005 See abstract; paragraphs [0052]-[0106]; and figures 1-6.	1-15
A	US 2005-0044339 A1 (KITRICK SHEETS) 24 February 2005 See abstract; paragraphs [0019]-[0037]; and figures 1-4.	1-15
A	US 2004-0044872 A1 (STEVEN, L. SCOTT) 04 March 2004 See abstract; paragraphs [0031]-[0038]; and figures 4-5.	1-15

☐ 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
16 March 2015 (16.03.2015)	**16 March 2015 (16.03.2015)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701,
Republic of Korea
Facsimile No. ++82 42 472 7140

Authorized officer
CHOI, Jeong Kwon
Telephone No. +82-42-481-8507

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2014/042563

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005-0273571 A1	08/12/2005	WO 2005-121969 A2	22/12/2005
		WO 2005-121969 A3	05/04/2007
US 2005-0039180 A1	17/02/2005	US 2012-054748 A1	01/03/2012
		US 2014-089923 A1	27/03/2014
		US 8544004 B2	24/09/2013
		US 8832692 B2	09/09/2014
US 2005-0044339 A1	24/02/2005	EP 1396790 A2	10/03/2004
		EP 1396790 A3	27/02/2008
		EP 1665054 A2	07/06/2006
		EP 2284711 A1	16/02/2011
		US 2004-0044872 A1	04/03/2004
		US 2005-0044128 A1	24/02/2005
		US 2005-0044340 A1	24/02/2005
		US 6922766 B2	26/07/2005
		US 7334110 B1	19/02/2008
		US 7437521 B1	14/10/2008
		US 7529906 B2	05/05/2009
		US 7543133 B1	02/06/2009
		US 7577816 B2	18/08/2009
		US 7743223 B2	22/06/2010
		US 8307194 B1	06/11/2012
		WO 2005-020088 A2	03/03/2005
		WO 2005-020088 A3	12/05/2005
US 2004-0044872 A1	04/03/2004	EP 1396790 A2	10/03/2004
		EP 1396790 A3	27/02/2008
		EP 1665054 A2	07/06/2006
		EP 2284711 A1	16/02/2011
		US 2005-0044128 A1	24/02/2005
		US 2005-0044339 A1	24/02/2005
		US 2005-0044340 A1	24/02/2005
		US 6922766 B2	26/07/2005
		US 7334110 B1	19/02/2008
		US 7437521 B1	14/10/2008
		US 7529906 B2	05/05/2009
		US 7543133 B1	02/06/2009
		US 7577816 B2	18/08/2009
		US 7743223 B2	22/06/2010
		US 8307194 B1	06/11/2012
		WO 2005-020088 A2	03/03/2005
		WO 2005-020088 A3	12/05/2005