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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 January 2016 (28.01.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/014049 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/US2014/047786

(22) International Filing Date:
23 July 2014 (23.07.2014)

(25) Filing Language: English

(26) Publication Language: English

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

(72) Inventor: **VAQUERO GONZALEZ, Luis Miguel**; HP Ltd., Longdown Avenue, Stoke Gifford Bristol BS34 8QZ (GB).

(74) Agents: **ESTEBAN, Michelle** et al.; Hewlett-Packard Company, Intellectual Property Administration, 3404 E. Harmony Road, Mail Stop 35, Fort Collins, Colorado 80528-9599 (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,

[Continued on next page]

(54) Title: RELAY TO PEER VIRTUAL MACHINE TRANSITION

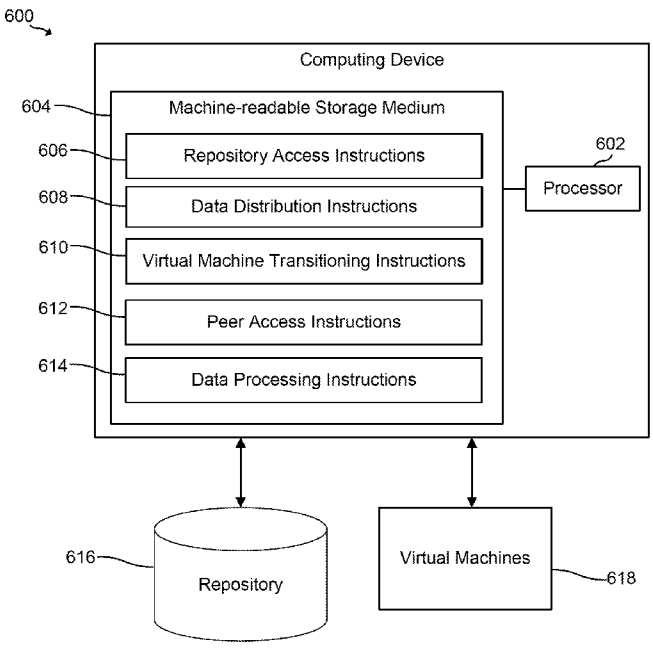


FIG. 6

(57) Abstract: Example implementations relate to relay to peer virtual machine transition. For example, a computing device may include a processor. The processor may access a partition of a data file from a data repository and distribute a portion of the partition to each peer virtual machine of a set of peer virtual machines associated with the computing device. The computing device may be a relay virtual machine relaying information to the set of peer virtual machines. The processor may transition the computing device from the relay virtual machine to a peer virtual machine. The processor may access data from another peer virtual machine and process the data.

WO 2016/014049 A1

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))*

RELAY TO PEER VIRTUAL MACHINE TRANSITION

BACKGROUND

[0001] Processing large data sets has become prevalent in research and business environments. Practitioners rely on tools to store and process increasingly larger amounts of data. Generally, a vast infrastructure capable of handling the large amounts of data may be required to store the data and carry out the processing. Cloud computing may provide a large-scale, on-demand infrastructure to accommodate varying workloads.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Some examples of the present application are described with respect to the following figures:

[0003] FIG. 1 is a block diagram of an example data management system;

[0004] FIG. 2 is a block diagram of an example file divided into partitions having some overlap with other partitions;

[0005] FIGS. 3-5 are schematic diagrams illustrating example arrangements of virtual machines transitioning from relay to peer virtual machines;

[0006] FIG. 6 is a block diagram of an example computing device for transitioning a virtual machine from a relay to a peer virtual machine; and

[0007] FIG. 7 is a flowchart illustrating an example method of transitioning a virtual machine from a relay to a peer virtual machine.

DETAILED DESCRIPTION

[0008] As described above, many entities may utilize tools to store and process large amounts of data. For big data deployment in a cloud computing environment, virtual machines may be created to perform the processing of data. Infrastructure as a Service (IaaS) cloud providers may provide such computation and storage services. These capabilities may be exposed through the use of an application programming interface (API) that may allow users to deploy virtual machines based on predefined virtual images using persistent storage services. When virtual machines are to be used to process data, the setup time of these virtual machines may include the amount of time for creation of the virtual machines, software configuration of the virtual machines, and provisioning of the virtual machines with data to be processed. While the task of populating freshly deployed virtual machines with data may be performed more easily for small data sets from a central repository, data transfer may become a bottleneck to performance as the data size grows. For example, transferring large amounts of data to a large number of virtual machines for processing may become a bottleneck if the data is to be sent from a central repository to each of the several virtual machines.

[0009] To accommodate for larger data sets, a hierarchical to peer-to-peer (P2P) data distribution may be used to reduce setup time on an on-demand cloud computing environment. This distribution technique couples dynamic topology with software configuration management tools (CMTs) to decrease setup time (e.g., virtual machine creation time, virtual machine software configuration time, virtual machine data loading time, etc.). To support a hierarchical to P2P data distribution, each virtual machine may be configured with suitable data transfer software tools (e.g., BitTorrent, Big Data applications, etc.).

[0010] When providing the virtual machines with the data to be processed, the data set from the central repository may be partitioned, and each partition may be assigned to a particular virtual machine for processing. Each partition of data may be distributed to its assigned virtual machine. The virtual machines that receive a partition may be a relay virtual machine that may

relay data to other virtual machines. A relay virtual machine may be a virtual machine that transfers portions of data to one or more peer virtual machines associated with the relay virtual machine based on a hierarchical relay tree. For example, a relay virtual machine may be a virtual machine that is a parent node to one or more peer virtual machines at child nodes. A peer virtual machine may be a virtual machine that may transfer data to and from other peer virtual machines using a P2P delivery approach.

[0011] When a relay virtual machine receives its assigned partition of data from the data repository, the relay virtual machine may distribute a portion of the partition to each of the peer virtual machines associated with the relay virtual machine. Once the partition is distributed among the peer virtual machines, the peer virtual machines may begin processing the data received. The relay virtual machine may transition to a peer virtual machine such that it may obtain unprocessed data from other peer virtual machines and may process the obtained data. As such, the network of virtual machines may transition from a hierarchical network of virtual machines to a P2P network of virtual machines during the processing time, allowing data to be processed by virtual machines as they become available to perform the processing.

[0012] Referring now to the figures, FIG. 1 is a block diagram of an example data management system 100. Data management system 100 may be one or more computing devices that include application-specific management engine 102, monitoring engine 104, automatic deployment engine 106, infrastructure abstraction engine 108, and processor 110. For example, data management system 100 may be, for example, one or more, or a combination of one or more, web-based servers, local area network servers, cloud-based servers, notebook computers, desktop computers, all-in-one systems, tablet computing devices, mobile phones, electronic book readers, or any other system for managing data in a cloud computing environment.

[0013] Application-specific management engine 102 may include processor executable instructions (e.g., executable by processor 110) and/or at least one electronic circuit that includes electronic components for managing various application-specific settings, configurations, and operations,

such as user-specified configuration settings, partitioning of data from a central repository, virtual machine configuration settings, and the like.

[0014] Applications-specific management engine 102 may receive user input specifying various configuration settings and may manage any preset configuration settings as well. For example, a user may provide parameters to size the virtual infrastructure (e.g., the number and size of slave nodes, the particular cloud computing provider to user, etc.). Applications-specific management engine 102 may also receive and manage any other information provided by a user, such as the topology of the virtual cluster, input files, output folder locations software configuration parameters (e.g., the total number of virtual machines, local directories where the data may be located, data file names, etc.), cloud provider credentials (e.g., username, password, token, etc.), and the like.

[0015] Applications-specific management engine 102 may manage data partitioning for data to be processed by one or more virtual machines. The applications-specific management engine 102 may operate as a centralized partitioning service that may run on top of where the original copy of the data is stored. For example, the centralized partitioning service may run on a logical volume on a storage area network attached to a partitioning virtual machine. When a new big data instantiation request arrives, this partitioning virtual machine may be created and configured to receive the data, partition the data, store a map of blocks to be transferred to the new virtual machines that may process the data, and return to the call controller. The central partitioning service of the applications-specific management engine 102 may enable dynamic loading of different partitioning strategies (e.g., binary partitioning, etc.). Data may be partitioned logically before transferring any data. An index may be created and stored to specify where each partition begins and ends. An example of a file partitioned by the centralized partitioning service of applications-specific management engine 102 is provided below in the explanation of FIG. 2.

[0016] Monitoring engine 104 may include processor executable instructions (e.g., executable by processor 110) and/or at least one electronic circuit that includes electronic components for monitoring and tracking the progress of data processing such that any corrective action may be taken to

keep the desired state of the resources. For example, monitoring engine 104 may track the progress of data processing to keep the processing running. In some examples, monitoring engine 104 may create additional virtual machines if monitoring engine 104 determines that other virtual machines have failed. Monitoring engine 104 may periodically pull monitoring information from virtual machines and/or push monitoring information to virtual machines.

[0017] Automatic deployment engine 106 may include processor executable instructions (e.g., executable by processor 110) and/or at least one electronic circuit that includes electronic components for performing the automatic deployment of the virtual infrastructure, automatic installation and configuration of the software installed in the virtual machines used to process the data, and the like. Automatic deployment engine 106 may create and initiate a predetermined number of virtual machines within the selected cloud computing provider. The virtual machines may be configured by a boot volume shared across the virtual machines that may be used to process the data. In some examples, the boot volume may contain a clean operating system (OS) installation with a CMT and a modified BitTorrent installed as daemons.

[0018] Automatic deployment engine 106 may also utilize an architecture similar to and/or compatible with SmartFrog, which is a pure P2P architecture with no central point of failure that enables fault-tolerant deployment and dynamic reconfiguration to change infrastructure and services at runtime. SmartFrog may also use late binding configuration information to configure services. The virtual machines may start the SmartFrog daemons during boot time. After the virtual machines have started, the SmartFrog daemon may dynamically provision the data processing services, following the received software installation and configuration instructions. Automatic deployment engine 106 may play the role of an orchestrator to make sure certain actions (e.g., SmartFrog setup, creation of the software distribution overlay, creation of the data distribution overlay, software downloads etc.) occur at the appropriate time. Upon completion of all the data transfers, the modified BitTorrent notifies SmartFrog, and SmartFrog initiates all the required big data processes on the virtual machines.

[0019] Automatic deployment engine 106 may manage the SmartFrog setup by downloading the application configuration file to a portion of the virtual machines created and initiating the P2P distribution on a software distribution overlay. The virtual machines operating as relay (e.g., master) virtual machines may receive the configuration file, which may include any relevant user-specified configuration settings as well as any relevant partitioning information, and the relay virtual machines may start seeding data to other virtual machines in a P2P manner.

[0020] Automatic deployment engine 106 may manage the creation of the software distribution overlay by downloading the software from the appropriate service catalogues, which may be repositories with various configuration files and software. These repositories may store the big data software packages to be copied and executed on the virtual machines as well as the configuration templates filled with the configuration parameters and copied into the virtual machines. The software configuration files and binaries may also be distributed in a P2P manner, which may allow for faster distribution and may prevent flash crowd calls against the repositories when a large number of virtual machines are deployed.

[0021] Automatic deployment engine 106 may manage the creation of the data distribution overlay. Once the download of the modified BitTorrent from the repositories has completed, automatic deployment engine 106 may use SmartFrog to configure BitTorrent with the portions of the data set to be processed by the BitTorrent host. Those portions may be specified by the centralized partitioning service of the applications-specific management engine 102. BitTorrent may be initiated by SmartFrog, and the data transfers may begin for the relay virtual machines, and for the peer virtual machines as the data is moved to the peer virtual machines.

[0022] Automatic deployment engine 106 may manage software downloads and installations. In some examples, the installed software may be configured, from an automatically generated configuration file, in parallel with the downloading of the data set to be processed. The automatically generated configuration file may be automatically created by the automatic deployment engine 106 using configuration parameters provided by the user.

In some examples, one of the virtual machines may be randomly chosen to be the master node providing the configuration file to the other slave nodes.

[0023] Infrastructure abstraction engine 108 may include processor executable instructions (e.g., executable by processor 110) and/or at least one electronic circuit that includes electronic components for homogenizing the APIs exposed by the different cloud computing infrastructure providers, providing a high-level view across different public and private infrastructure providers.

[0024] Processor(s) 110 may be one or more, or a combination of one or more, central processing units (CPUs), semiconductor-based microprocessors, and/or other hardware devices suitable for retrieval and execution of instructions to perform at least a portion of the functions of application-specific management engine 102, monitoring engine 104, automatic deployment engine 106, infrastructure abstraction engine 108, and processor 110, as described above.

[0025] FIG. 2 is a block diagram of an example file 200 divided into partitions having some overlap with other partitions. The example original file 200 may be partitioned, using the centralized partitioning service of applications-specific management engine 102 of FIG. 1, into four parts: Part 1, Part 2, Part 3, and Part 4. After partitioning the file, a portion of each part is overlapped with other parts. For example, a portion of Part 1 is added to Part 2, a portion of Part 2 is added to Part 1 and Part 3, a portion of Part 3 is added to Part 2 and Part 4, and a portion of Part 4 is added to Part 3. The added overlapped portions of each partition, which may originate from other partitions, may be ignored by the data set loader on each of the virtual machines but may be used to seed other peer virtual machines that may use the data. The amount of overlap may be optimized and/or selected by the centralized partitioning service of applications-specific management engine 102 of FIG. 1. The higher the amount of overlap, the higher the consumption of bandwidth and the lower the transfer time (e.g., assuming the actual maximum capacity of the underlying network fabric has not been reached). However, if the overlap is not big enough, then some of the partitions may only be stored in one of the initial relay virtual machines accessing the central repository. In some examples, a full overlap may not be used and instead

some portions of data may be located only in the central repository. Access to those portions may be randomized to reduce the likelihood of a flash crowd effect. In some examples, the overlapped partitions may be based on a binary partition of the original file 200. For example, the number of bytes of the original file 200 may be determined, and the number of bytes may be evenly divided among the number of desired partitions of data. In some examples, the partitions may be based on the structure of the original file 200 (e.g., if the original file 200 is a graph, create partitions based on the graph). In some examples, the number of partitions created may be based on the number of relay virtual machines created.

[0026] Once logical partitions are ready for distribution across peer virtual machines, the distribution of data may occur such that data transfer redundancy is minimized and transfer rates are maximized. A torrent controller, which may be a lightweight hypertext transfer protocol daemon (httpd) that hosts an announce path onto which BitTorrent clients update their state, may be used to achieve this optimization. The central repository may be used to provide the files to be deployed onto the virtual machines.

[0027] The centralized partitioning service may decide how the original file is split across virtual machines. After obtaining the global partitioning index from the centralized partitioning service, each partition of the file may be placed into a directory, and a new torrent is created for each directory. This directory may then be compressed, packaged, and transformed into a .torrent file, which may be a file with basic hash information about the compressed directory. The BitTorrent tracker may keep track of which .torrent files are being distributed as they are being distributed.

[0028] The CMT may set a predefined number of virtual machines as relay virtual machines. In some examples, this predefined number of relay virtual machines may be set by using a heuristic value (e.g., $log(N)^3$, where N is the total number of virtual machines). Relay virtual machines may connect to the central repository to obtain data that may have been previously uploaded. The peer virtual machines may receive blank information from the BitTorrent tracker until the relay virtual machines receive data. The peer virtual machines may connect to their assigned relay virtual machine, and these relay virtual machines may relay data to their associated peer virtual

machines, which may prevent an initial avalanche that may occur if all peer virtual machines accessed the central repository while initializing the download (e.g., flash crowd effect).

[0029] The peer virtual machines may receive portions of the partitions and may distribute the portions among other peer virtual machines. Once a peer virtual machine is finished downloading its assigned portion, it may continue seeding for a configurable amount of time (e.g., depending on the size of the data downloaded) to prevent a hotspot effect on its associated relay virtual machine.

[0030] Because the number of initial relay virtual machines is smaller than the total number of virtual machines, the relay virtual machines may receive data from one or more partitions to serve as relay virtual machines for a number of peer virtual machines. Once peer virtual machines begin receiving portions of data and make the portions available to other peer virtual machines, relay virtual machines may be automatically reconfigured by the data management system 100 of FIG. 1 as peer virtual machines. If the partitioning algorithm leave some portions of data without replication and no relay virtual machines receive data, the peer virtual machines may randomly connect to the central repository when they no longer have data to be processed and may download data for processing.

[0031] This reconfiguration from relay virtual machines to peer virtual machines may be expressed as a Markov chain model, which may have as state variables the populations of initial relay virtual machines and peer virtual machines in the system behaving in the manner as follows. Peer virtual machines may arrive as a Poisson process of rate λ to their assigned relay virtual machines, stay in a peer virtual machine queue until completing the download, and then stay in a relay virtual machine queue for an exponential time of parameter γ, where the mean upload time for the file is $1/\mu$, which may lead to a continuous time Markov chain. Assuming the mean upload time for a portion of data is $1/\mu$, this may be capped by a particular data transfer rate (e.g., 10 MB/s limit) that may be imposed to avoid crashing the network. In some examples, λ may be controlled by the data management system 100 of FIG. 1 such that the rate of arrival of peer virtual machines may be controlled at the beginning. The tracker may be modified to filter out responses from

peer virtual machines during the initial setup stages. The particular data transfer rate may indirectly limit the value of λ both initially and through the data transfer.

[0032] In some examples, the number of initial relay virtual machines accessing the central repository may be constrained and controlled to prevent the flash crowd effect. Other nodes may be capped peer virtual machines (e.g., leecher virtual machines) such that these nodes are prevented from establishing any connection until their assigned initial relay virtual machine receives data from the central repository. Thus, during the download of the initial data, $x(t)$ may denote the number of capped peer virtual machines at time t, and $y(t)$ may denote the number of relay virtual machines. As such, the total upload capacity in the system in partitions per second becomes $\Omega_{up} = \mu (y + x)$. Because $x(0) = 0$ and $y(0) = 0$, since the initial relay virtual machines are not uploading anything, $\Omega_{up} = 0$.

[0033] Additionally, Ω_{down} may denote the total download rate of the system in partitions per second as $\Omega_{down} = \mu log(N)^3$, where N is the total number of virtual machines.

[0034] At this point, there may be a transient period where initial relay virtual machines start to supply peer virtual machines with portions of data, and peer virtual machines that finish downloading portions start to seed themselves as follows:

$$dx/dt = \lambda - \Omega_{down}(x, y)$$

$$dy/dt = \Omega_{down}(x, y) - \gamma(y)$$

where $\Omega_{down}(x, y) = min\{\Omega_{up}, cx\}$ states that Ω_{down} is constrained by either the available upload capacity or the maximum download rate per peer virtual machine, defined as c.

[0035] After this transient period, the system reaches a dynamic equilibrium characterized by a total upload capacity in the system in partitions per second, as $\Omega_{up} = \mu (y + x)$. Because $x(t) = M$ and $y(0) = 0$, since the virtual machines are now all peer virtual machines, $\Omega_{up} = \mu M$. At an equilibrium of dx/dt and dy/dt, the number of peer virtual machines is $y^* = \lambda / \gamma$. After some time, the number of relay virtual machines may approach zero after the initial setup period, independently of the value of λ.

[0036] FIGS. 3-5 are schematic diagrams illustrating example arrangements of virtual machines transitioning from relay to peer virtual machines. FIGS. 3-5 show how the topology of the distribution network changes from a hierarchical structure to a P2P structure.

[0037] In FIG. 3, arrangement 300 of virtual machines 2-101 shows the initial hierarchical, tree-like infrastructure of virtual machines 2-101. Virtual machines 2-101 may process data originating from central repository 1. The hierarchical arrangement 300 shows virtual machines 2-9 being relay virtual machines capable of accessing data directly from central repository 1. Virtual machines 10-101 may be peer virtual machines that may each access data from their associated relay virtual machine. For example, peer virtual machine 46 may be associated with relay virtual machine 5 and may receive data from relay virtual machine 5. This arrangement 300 may prevent the flash crowd effect of a large number of virtual machines 2-101 accessing central repository 1 at once.

[0038] In FIG. 4, arrangement 400 of virtual machines 2-101 shows the evolution to a more decentralized P2P structure after a certain amount of time passes from the initial arrangement 300 of FIG. 3. This evolution may increase throughput of data between virtual machines 2-101.

[0039] In FIG. 5, arrangement 500 of virtual machines 2-101 shows the evolution to a decentralized P2P structure after a certain amount of time passes from the arrangement 400 of FIG. 4. This evolution may increase throughput of data between virtual machines 2-101. In some examples, peer virtual machines may coordinate with data management system 100 of FIG. 1 when no other peer virtual machines has data. In this case, the peer virtual machine may access data directly from the central repository 1. In some examples, access to central repository 1 may be randomized by the data management system 100 upon starting BitTorrent.

[0040] FIG. 6 is a block diagram of an example computing device 600 for transitioning a virtual machine from a relay to a peer virtual machine. Computing device 600 may be a virtual machine that may operate as a relay virtual machine and/or a peer virtual machine.

[0041] Computing device 600 may be, for example, a web-based server, a local area network server, a cloud-based server, a notebook computer, a

desktop computer, an all-in-one system, a tablet computing device, a mobile phone, an electronic book reader, or any other electronic device suitable for transitioning computing device 600 from a relay to a peer virtual machine. Computing device 600 may include a processor 602 and a machine-readable storage medium 604. Computing device 600 may be in communication with one or more other virtual machines 618. Computing device 600 may be capable of relaying data from repository 616 to the other virtual machines 618 and processing data from repository 616.

[0042] Processor 602 may be a CPU, a semiconductor-based microprocessor, and/or other hardware devices suitable for retrieval and execution of instructions stored in machine-readable storage medium 604. Processor 602 may fetch, decode, and execute instructions 606, 608, 610, 612, and 614 to control a process of transitioning computing device 600 from a relay to a peer virtual machine. As an alternative or in addition to retrieving and executing instructions, processor 602 may include at least one electronic circuit that includes electronic components for performing the functionality of instructions 606, 608, 610, 612, 614, or a combination thereof.

[0043] Machine-readable storage medium 604 may be any electronic, magnetic, optical, or other physical storage device that contains or stores executable instructions. Thus, machine-readable storage medium 604 may be, for example, Random Access Memory (RAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), a storage device, an optical disc, and the like. In some examples, machine-readable storage medium 604 may be a non-transitory storage medium, where the term “non-transitory” does not encompass transitory propagating signals. As described in detail below, machine-readable storage medium 604 may be encoded with a series of processor executable instructions 606, 608, 610, 612, and 614 for accessing a partition of a data file from repository 616, distributing a portion of the partition to each peer virtual machine of a set of peer virtual machines (e.g., one or more of virtual machines 618) associated with computing device 600, transitioning computing device 600 from a relay virtual machine to a peer virtual machine, accessing data from another peer virtual machine (e.g., from one of virtual machines 618), and processing the data.

[0044] Repository access instructions 606 may manage and control access to repository 616 for data to be relayed and/or processed by computing device 600. For example, repository access instructions 606 may manage and control access to partitions of data stored in repository 616.

[0045] Data distribution instructions 608 may manage and control the distribution of portions of data accessed from repository 616. For example, data distribution instructions 608 may manage and control the distribution of data to one or more peer virtual machines associated with computing device 600.

[0046] Virtual machine transitioning instructions 610 may manage and control the transition of computing device 600 from a relay virtual machine to a peer virtual machine. For example, virtual machine transitioning instructions 610 may coordinate with data management system 100 of FIG. 1 to transition computing device 600 from a relay virtual machine to a peer virtual machine after data accessed from repository 616 has been distributed to the associated peer virtual machines.

[0047] Peer access instructions 612 may manage and control access to other peer virtual machines processing data from repository 616. For example, peer access instructions 612 may manage and control access to other peer virtual machines (e.g., at least a portion of virtual machines 618) to send and/or receive data to be processed by computing device 600 once peer computing device 100 has transitioned to a peer virtual machine. Peer virtual machines may make their data available to other peer virtual machines that may be available for processing (e.g., peer virtual machines with no data left to process) such that another peer virtual machine may download data to aid in the processing of the data.

[0048] Data processing instructions 614 may manage and control the processing of data originating from repository 616. For example, data processing instructions 614 may manage and control the processing of data received from other peer virtual machines, such as a portion of virtual machines 618.

[0049] Repository 616 may be a central repository storing data to be processed. For example, repository 616 may be a cloud computing storage system for storing large amounts of data.

[0050] Virtual machines 618 may be one or more virtual machines processing data from repository 616. For example, virtual machines 618 may include peer virtual machines associated with computing device 600 when computing device 600 operates as a relay virtual machine and may include other peer virtual machines that computing device 600 may access when computing device 600 operates as a peer virtual machine.

[0051] FIG. 7 is a flowchart illustrating an example method 700 of transitioning a virtual machine from a relay to a peer virtual machine. Method 700 may be implemented using computing device 600 of FIG. 6.

[0052] Method 700 includes, at 702, accessing, by a computing device (e.g., computing device 600), a partition of a data file from a data repository (e.g., repository 616). The computing device may operate as a relay virtual machine to relay data to peer virtual machines associated with the computing device.

[0053] Method 700 also includes, at 704, distributing a portion of the partition of the data file to each peer virtual machine of a set of peer virtual machines associated with the computing device operating as a relay virtual machine.

[0054] Method 700 also includes, at 706, transitioning the computing device from the relay virtual machine to a peer virtual machine. The computing device may transition to a peer virtual machine after the portions of the partition of the data file have been distributed to the associated peer virtual machines.

[0055] Method 700 also includes, at 708, accessing data from another peer virtual machine. For example, the computing device operating as a peer virtual machine may access data from other peer virtual machines with unprocessed data.

[0056] Method 700 also includes, at 710, processing the data accessed from the other peer virtual machines.

[0057] Examples provided herein (e.g., methods) may be implemented in hardware, software, or a combination of both. Example systems may include a controller/processor and memory resources for executing instructions stored

in a tangible non-transitory medium (e.g., volatile memory, non-volatile memory, and/or machine-readable media). Non-transitory machine-readable media can be tangible and have machine-readable instructions stored thereon that are executable by a processor to implement examples according to the present disclosure.

[0058] An example system can include and/or receive a tangible non-transitory machine-readable medium storing a set of machine-readable instructions (e.g., software). As used herein, the controller/processor can include one or a plurality of processors such as in a parallel processing system. The memory can include memory addressable by the processor for execution of machine-readable instructions. The machine-readable medium can include volatile and/or non-volatile memory such as a random access memory ("RAM"), magnetic memory such as a hard disk, floppy disk, and/or tape memory, a solid state drive ("SSD"), flash memory, phase change memory, and so on.

Claims

What is claimed is:

1. A computing device comprising:

a processor to:

access a partition of a data file from a data repository;

distribute a portion of the partition to each peer virtual machine of a set of peer virtual machines associated with the computing device, the computing device being a relay virtual machine relaying information to the set of peer virtual machines;

transition the computing device from the relay virtual machine to a peer virtual machine;

access data from another peer virtual machine; and

process the data.

2. The computing device of claim 1, wherein the data accessed by the computing device is capable of being accessed by other peer virtual machines.

3. The computing device of claim 1, wherein the processor is further to:

identify a particular peer virtual machine with additional data to be processed;

access at least a portion of the additional data; and

process the at least a portion of the additional data.

4. The computing device of claim 1, wherein the processor is further to:

access additional data from the data repository; and

process the additional data.

5. The computing device of claim 1, wherein the processor is further to publish a status indicating the processing of the data is complete.

6. A method comprising:

receiving, by a computing device, a partition of a data file from a data repository;

sending, by the computing device, a portion of the partition to each peer virtual machine of a set of peer virtual machines associated with the computing device, the computing device being a relay virtual machine relaying information to the set of peer virtual machines;

reconfiguring, by the computing device, the computing device from the relay virtual machine to a peer virtual machine;

accessing, by the computing device, data from another peer virtual machine;

processing, by the computing device, the data; and

providing, by the computing device, the processed data.

7. The method of claim 6, wherein the data accessed by the computing device is capable of being accessed by other peer virtual machines.

8. The method of claim 6, further comprising:

identifying, by the computing device, a particular peer virtual machine with additional data to be processed;

accessing, by the computing device, at least a portion of the additional data; and

processing, by the computing device, the at least a portion of the additional data.

9. The method of claim 6, further comprising:

accessing, by the computing device, additional data from the data repository; and

processing, by the computing device, the additional data.

10. The method of claim 6, further comprising:

publishing a status indicating the processing of the data is complete.

11. A non-transitory machine-readable storage medium storing instructions that, if executed by at least one processor of a computing device, cause the computing device to:

access a partition of a data file from a data repository;

send a portion of the partition to each peer virtual machine of a set of peer virtual machines associated with the computing device, the computing device being a relay virtual machine relaying information to the set of peer virtual machines;

change the computing device from the relay virtual machine to a peer virtual machine;

obtain data from another peer virtual machine; and

process the data;

12. The non-transitory machine-readable storage medium of claim 11, wherein the data obtained is capable of being accessed by other peer virtual machines.

13. The non-transitory machine-readable storage medium of claim 11, wherein the instructions, if executed by the at least one processor, further cause the computing device to:

identify a particular peer virtual machine with additional data to be processed;

access at least a portion of the additional data; and

process the at least a portion of the additional data.

14. The non-transitory machine-readable storage medium of claim 11, wherein the instructions, if executed by the at least one processor, further cause the computing device to:

access additional data from the data repository; and

process the additional data.

15. The non-transitory machine-readable storage medium of claim 11, wherein the instructions, if executed by the at least one processor, further cause the computing device to publish a status indicating the processing of the data is complete.

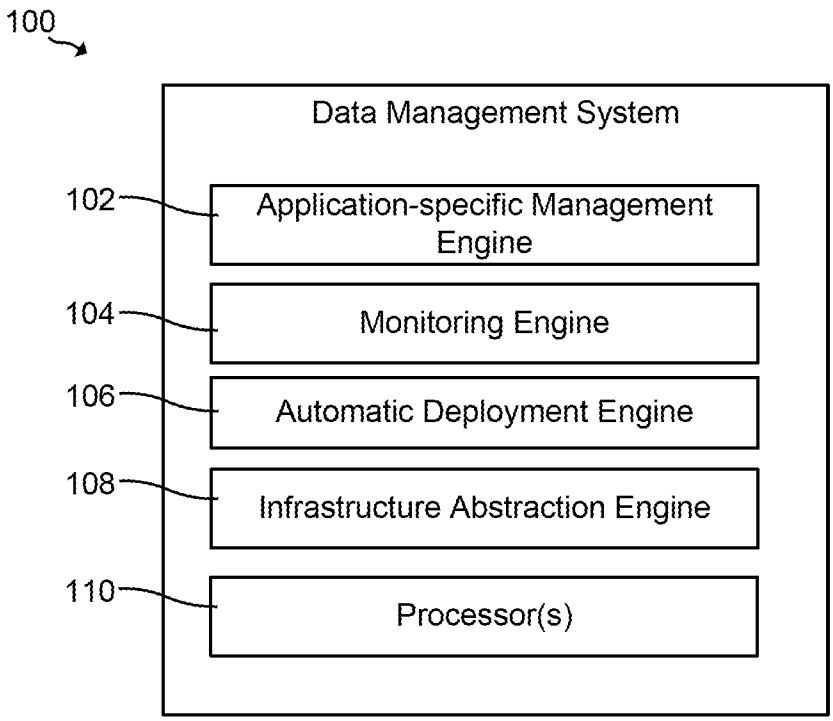

100
Data Management System
102
Application-specific Management Engine
104
Monitoring Engine
106
Automatic Deployment Engine
108
Infrastructure Abstraction Engine
110
Processor(s)

FIG. 1

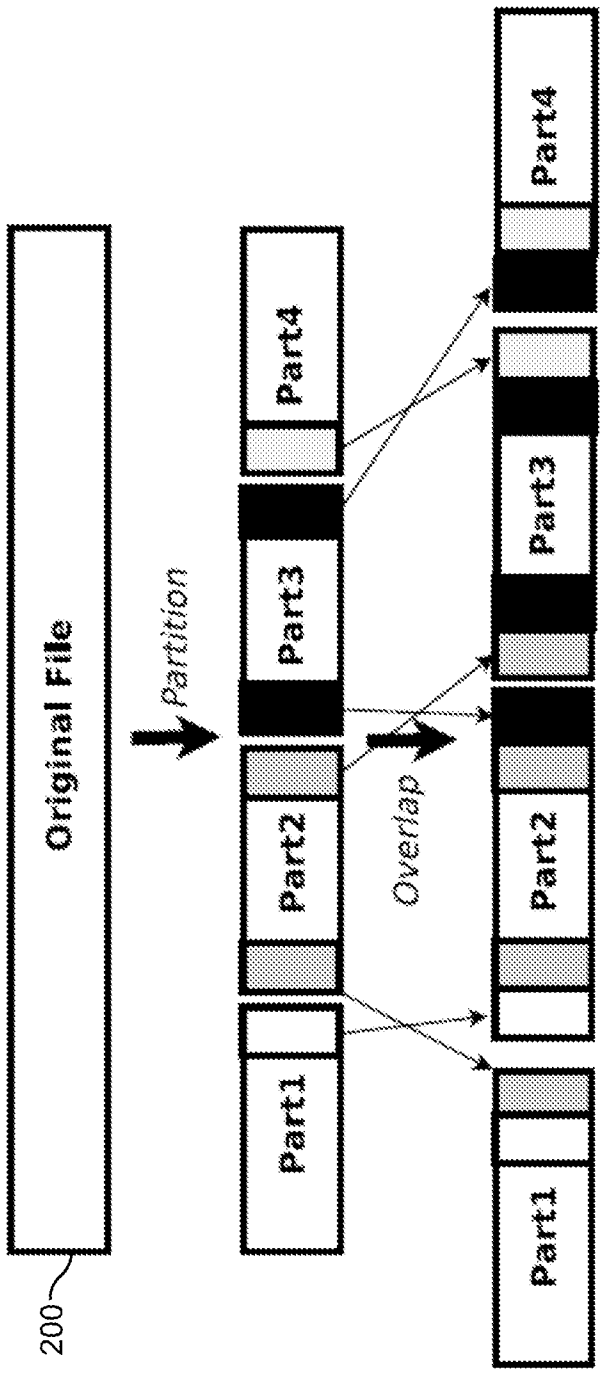

Original File
200
Partition
Part1
Part2
Part3
Part4
Overlap
Part1
Part2
Part3
Part4

FIG. 2

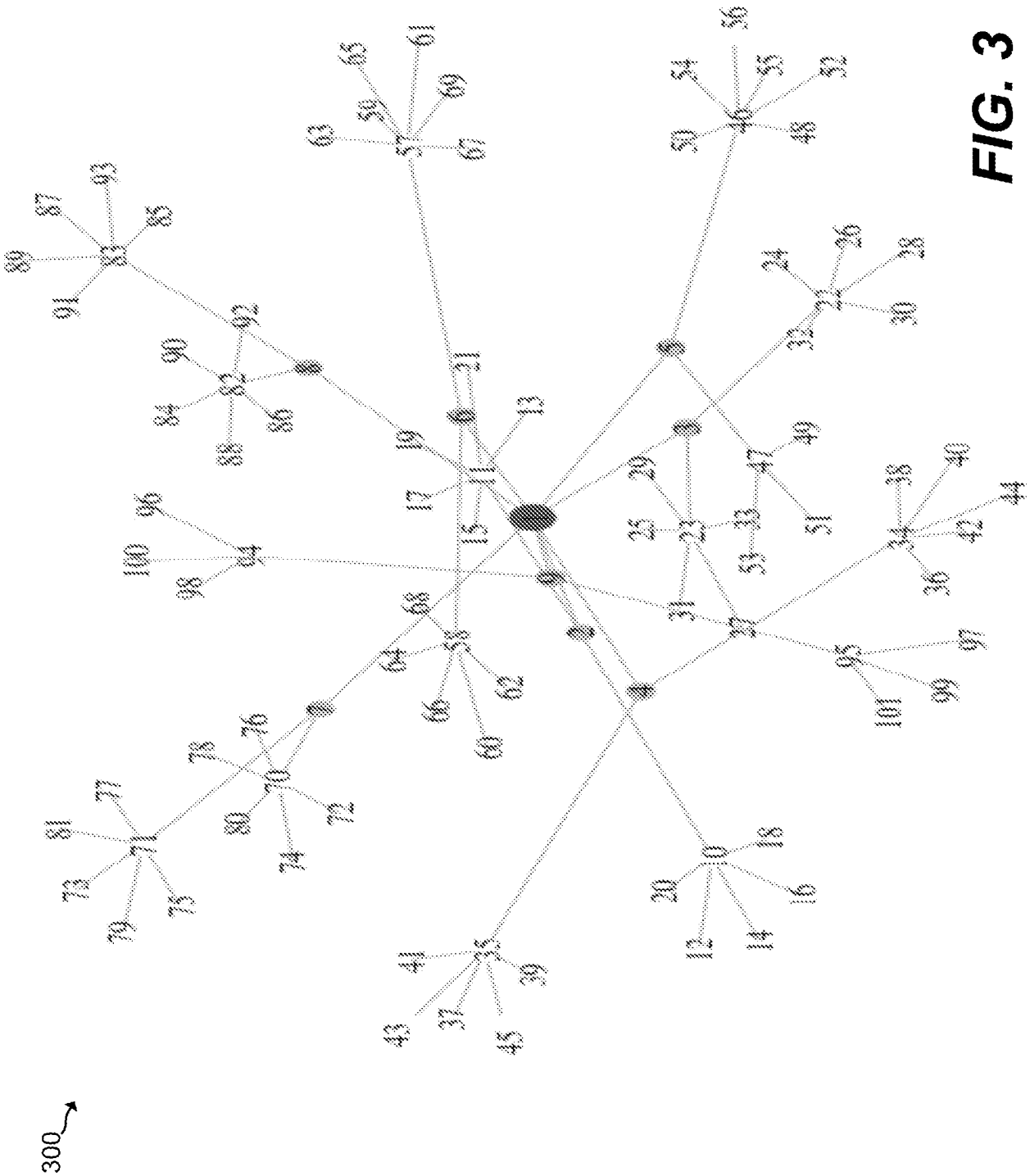


FIG. 3

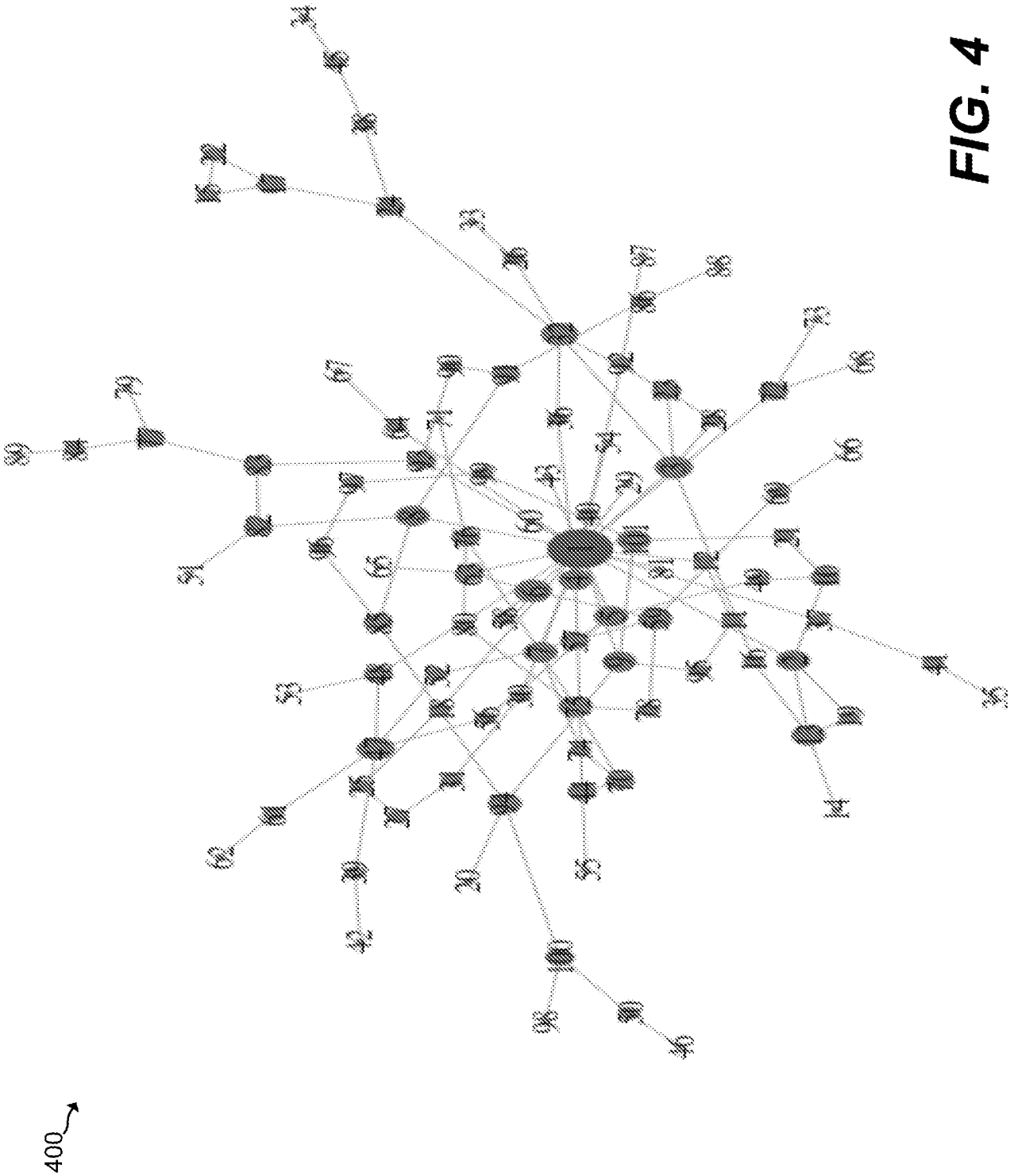
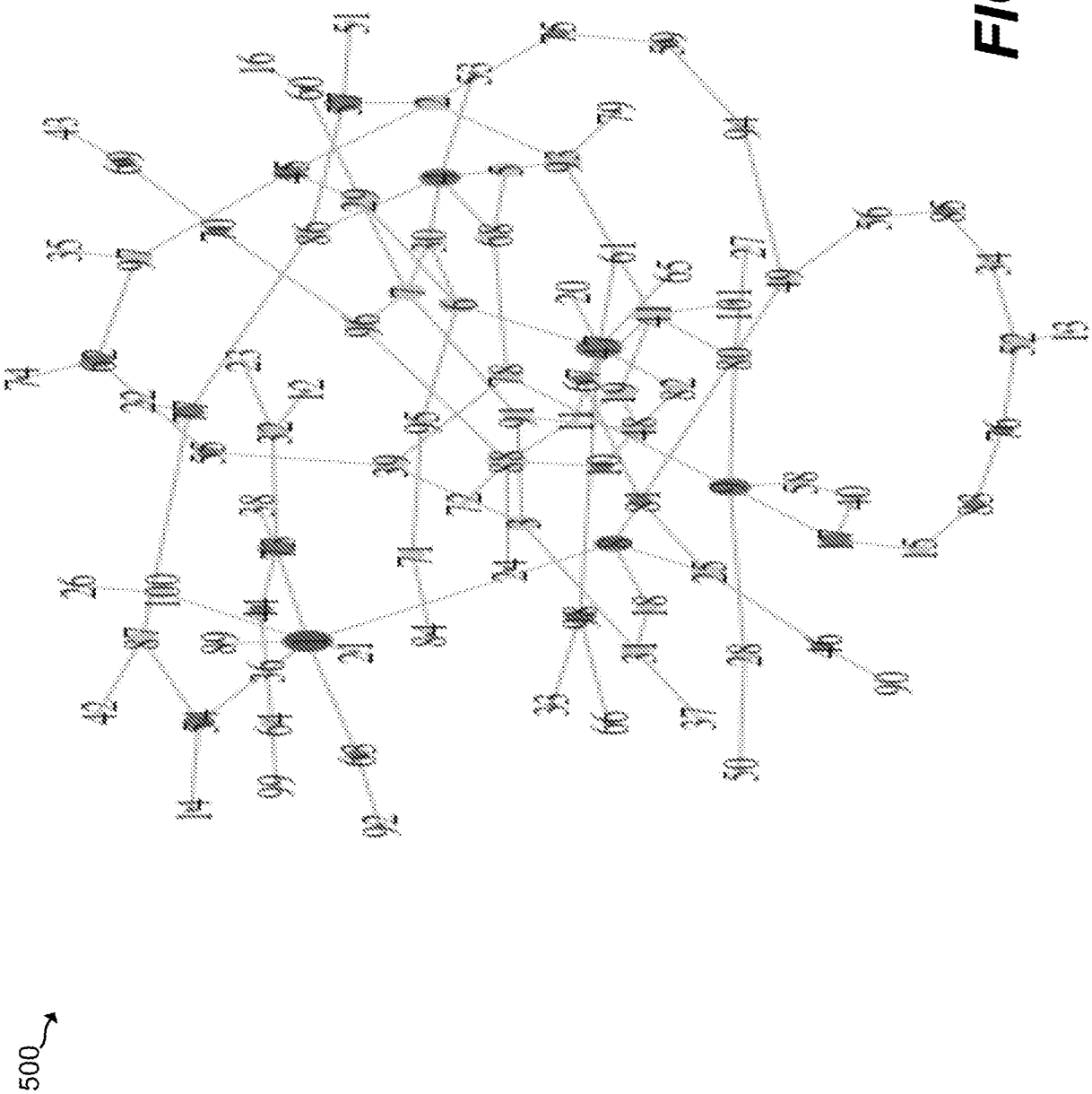


FIG. 4

FIG. 5



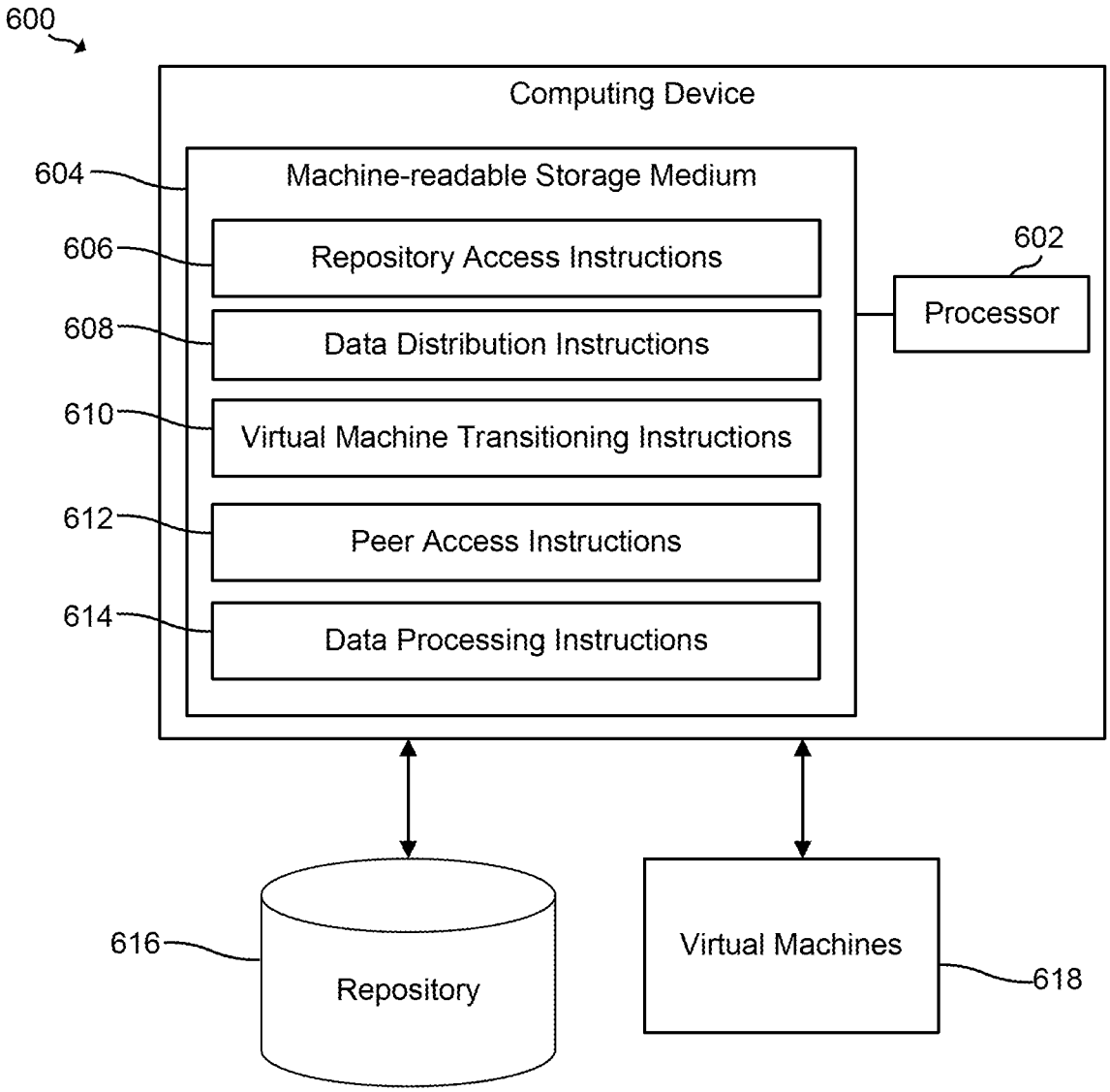

600
Computing Device
604
Machine-readable Storage Medium
606
Repository Access Instructions
608
Data Distribution Instructions
610
Virtual Machine Transitioning Instructions
612
Peer Access Instructions
614
Data Processing Instructions
602
Processor
616
Repository
Virtual Machines
618

FIG. 6

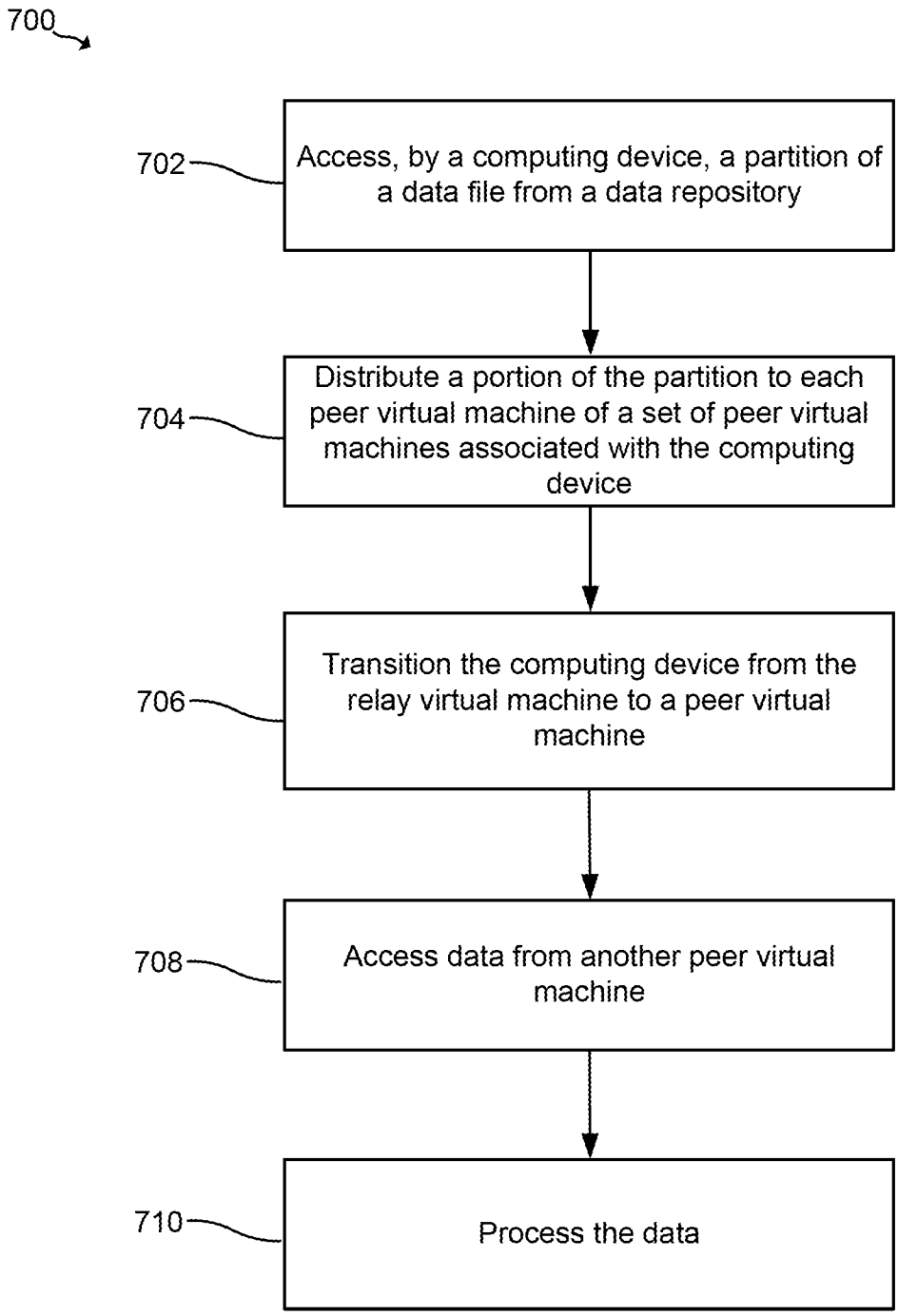

700
702
Access, by a computing device, a partition of a data file from a data repository
704
Distribute a portion of the partition to each peer virtual machine of a set of peer virtual machines associated with the computing device
706
Transition the computing device from the relay virtual machine to a peer virtual machine
708
Access data from another peer virtual machine
710
Process the data

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/047786

A. CLASSIFICATION OF SUBJECT MATTER

G06F 15/16(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 15/16; H04L 9/08; G06F 15/173; G06F 9/455

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: relay, peer, virtual machine, VM, transit*

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 8677358 B2 (JAMES B. EVANS et al.) 18 March 2014 See abstract; Col.5 line 1 - Col.8 line52, Col.15 line20 - Col.16 line11, Col.19 line 63 - Col.20 line 16; figures 5, 9B, 10, 19.	1,2,4,5
A		3,6-15
Y	EP 2651072 A2 (SECURITY FIRST CORP.) 16 October 2013 See abstract; paragraphs [0409]-[0411], [0488]-[0490]; figures 26,27,38.	1,2,4,5
A		3,6-15
A	US 2009-0133017 A1 (BOOGERT KEVIN M. et al.) 21 May 2009 See abstract; paragraphs [0021]-[0030]; figure 2.	1-15
A	US 8321862 B2 (SUMEDHA K. SWAMY et al.) 27 November 2012 See abstract; Col.6 line 44 - Col.7 line 40; figure 4.	1-15
A	US 8560663 B2 (STEPHAN BAUCKE et al.) 15 October 2013 See abstract; Col.13 line 39 - Col.15 line 22; figure 9.	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
23 March 2015 (23.03.2015)	**23 March 2015 (23.03.2015)**

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
KYUNG, Youn Jeong
Telephone No. +82-42-481-3452

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

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2014/047786

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 8677358 B2	18/03/2014	US 2011-0107331 A1	05/05/2011
		US 2011-0167472 A1	07/07/2011
		US 2011-0167473 A1	07/07/2011
		US 8621460 B2	31/12/2013
		WO 2011-053976 A1	05/05/2011
EP 2651072 A2	16/10/2013	AU 2011-305569 A1	11/04/2013
		CA 2812986 A1	29/03/2012
		CN 103609059 A	26/02/2014
		EP 2619939 A2	31/07/2013
		EP 2651072 A3	23/10/2013
		US 2012-072723 A1	22/03/2012
		US 2014-298012 A1	02/10/2014
		US 8769270 B2	01/07/2014
		WO 2012-040231 A2	29/03/2012
		WO 2012-040231 A3	28/06/2012
		WO 2012-040231 A9	29/03/2012
		WO 2012-040231 A9	20/09/2012
US 2009-0133017 A1	21/05/2009	US 8930945 B2	06/01/2015
US 8321862 B2	27/11/2012	US 2010-242045 A1	23/09/2010
US 8560663 B2	15/10/2013	EP 2579527 A1	10/04/2013
		US 2013-0086236 A1	04/04/2013