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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 July 2014 (03.07.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/105984 A2

(51) International Patent Classification:
G06F 11/14 (2006.01) *G06F 9/455* (2006.01)

(21) International Application Number:
PCT/US2013/077865

(22) International Filing Date:
26 December 2013 (26.12.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/728,403 27 December 2012 (27.12.2012) US

(71) Applicant: **NUTANIX, INC.** [US/US]; 1735 Technology Drive, Suite 575, San Jose, California 95110 (US).

(72) Inventors: **GILL, Binny Sher**; 1073 Mckay Drive, San Jose, California 95131 (US). **BYRNE, Brian**; 1700 North 1st Street, Apt. 130, San Jose, California 95112 (US). **ARON, Mohit**; 1880 Fallen Leaf Lane, Los Altos, California 94024 (US).

(74) Agent: **HSU, Frederick**; Vista Ip Law Group, LLP, 2160 Lundy Avenue, Suite 230, San Jose, California 95131 (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.

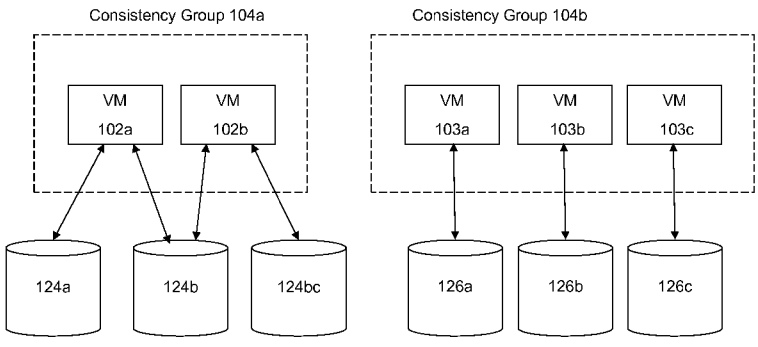
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, 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).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: METHOD AND SYSTEM FOR IMPLEMENTING CONSISTENCY GROUPS WITH VIRTUAL MACHINES

FIG. 1



(57) Abstract: Disclosed is an approach for implementing disaster recovery for virtual machines. Consistency groups are implemented for virtual machines, where the consistency group link together two or more VMs. The consistency group includes any set of VMs which need to be managed on a consistent basis in the event of a disaster recovery scenario.

WO 2014/105984 A2

METHOD AND SYSTEM FOR IMPLEMENTING CONSISTENCY GROUPS WITH VIRTUAL MACHINES

Field

[0001] This disclosure concerns virtual machine and storage technology.

Background

[0002] There are many kinds of architectures that can be used to implement storage systems. Traditionally, storage for a computing system is implemented using directly attached or integrated storage, such as hard disk drives that are commonly integrated into personal computers. Distributed storage architectures are also widely used, to allow a computer to access and store data on networked based storage devices.

[0003] Modern computing systems may also implement storage in the context of virtualization environments. A virtualization environment contains one or more "virtual machines" or "VMs", which are software-based implementation of a machine in an environment in which the hardware resources of a real computer (e.g., CPU, memory, storage, etc.) are virtualized or transformed into the underlying support for the fully functional virtual machine that can run its own operating system and applications on the underlying physical resources just like a real computer. By encapsulating an entire machine, including CPU, memory, operating system, storage devices, and network devices, a virtual machine is completely compatible with most standard operating systems, applications, and device drivers. Virtualization allows

one to run multiple virtual machines on a single physical machine, with each virtual machine sharing the resources of that one physical computer across multiple environments. Different virtual machines can run different operating systems and multiple applications on the same physical computer.

[0004] One reason for the broad adoption of virtualization in modern business and computing environments is because of the resource utilization advantages provided by virtual machines. Without virtualization, if a physical machine is limited to a single dedicated operating system, then during periods of inactivity by the dedicated operating system the physical machine is not utilized to perform useful work. This is wasteful and inefficient if there are users on other physical machines which are currently waiting for computing resources. To address this problem, virtualization allows multiple VMs to share the underlying physical resources so that during periods of inactivity by one VM, other VMs can take advantage of the resource availability to process workloads. This can produce great efficiencies for the utilization of physical devices, and can result in reduced redundancies and better resource cost management.

[0005] Storage devices comprise one type of a physical resource that can be managed and utilized in a virtualization environment. A set of one or more virtual disks may be implemented to allow virtual storage of data on behalf of one or more clients, such as client computers, systems, applications, or virtual machines, where the virtual disk (or "vdisk") is actually a logical representation of storage space compiled from one or more physical underlying storage devices. When the client issues a write request or read request in a virtualized system, that request is actually issued to a virtualized storage device.

[0006] The topic addressed by the present disclosure pertains to disaster recovery scenarios involving VMs. Modern organizations have come to the realization that processes and procedures need to be put into place to address the possibility of disasters and failures, both natural or human-induced, that may affect the computing equipment used by the organization. As computing and information technology systems have become increasingly critical to the operations of an organization, the importance of ensuring the continued operation of those systems has increased.

SUMMARY

[0007] Embodiments of the present invention provide an improved approach for implementing disaster recovery for VMs.

[0008] In accordance with some embodiments, a method for implementing a consistency group, involves locking resources associated with the members of a consistency group, wherein the members of the consistency group comprises a group of related virtual machines, generating a snapshot of the resources associated with the members of the consistency group, and releasing locks on the resource after the snapshot has been generated.

[0009] In one or more embodiments, an application associated with the resources does not have a native functionality to generate the snapshot.

[0010] In one or more embodiments, an application associated with the resources is not modified to include functionality to implement the consistency group.

[0011] In one or more embodiments, the snapshot comprises a copy of one or more virtual disks associated with the resources.

[0012] In one or more embodiments, the snapshot comprises identification of one or more identifier numbers associated with the copy of the one or more virtual disks associated with the resources.

[0013] In one or more embodiments, copy-on-write functionality is used to create the copy of the one or more virtual disks associated with the resources, wherein the one or more identifier numbers is incremented upon making the copy.

[0014] In one or more embodiments, metadata is maintained to track membership in the consistency group.

[0015] In one or more embodiments, the snapshot is generated at a time period using at least one of: a regular basis or an ad hoc basis.

[0016] In one or more embodiments, the method further comprises identifying the consistency group to restore, identifying the snapshot for the consistency group, and initiating the group of related virtual machines using the snapshot.

[0017] In one or more embodiments, the snapshot is associated with the group of related virtual machines.

[0018] In one or more embodiments, disaster recovery is performed.

[0019] Further details of aspects, objects, and advantages of the invention are described below in the detailed description, drawings, and claims. Both the foregoing general description and the following detailed description are exemplary and explanatory, and are not intended to be limiting as to the scope of the invention.

Brief Description of the Drawings

[0020] The drawings illustrate the design and utility of embodiments of the present invention, in which similar elements are referred to by common reference numerals. In order to better appreciate the advantages and objects of embodiments of the invention, reference should be made to the accompanying drawings. However, the drawings depict only certain embodiments of the invention, and should not be taken as limiting the scope of the invention.

[0021] Fig. 1 illustrates VM-based consistency groups according to some embodiments of the invention.

[0022] Fig. 2 shows a flowchart of an approach for VM-based consistency groups according to some embodiments of the invention.

[0023] Fig. 3 shows a flowchart of an approach for recovering VM-based consistency groups according to some embodiments of the invention.

[0024] Figs. 4A-E illustrate implementation of VM-based consistency groups according to some embodiments of the invention.

[0025] Fig. 5 is a block diagram of a computing system suitable for implementing an embodiment of the present invention.

Detailed Description of the Embodiments of the Invention

[0026] Embodiments of the present invention provide an improved approach for implementing disaster recovery for VMs. In some embodiments of the invention, consistency groups are implemented for virtual machines, where the consistency group links together two or more VMs. The consistency group includes any set of VMs which need to be managed on a consistent basis in the event of a disaster recovery scenario.

[0027] Fig. 1 illustrates example consistency groups 104a and 104b according to some embodiments of the invention. Each consistency group 104a and 104b includes a group of VMs that need to be maintained to be crash consistent. This means that the state of the VMs in a respective consistency group must be maintained on a consistent basis across the member VMs after disaster recovery has occurred. The VMs in the consistency group are therefore snapshotted at the same point in time as if the snapshots occurred instantaneously.

[0028] There may be any number of VMs in a consistency group as required to maintain consistency across sets of related VMs. As illustratively shown in the figure, consistency group 104a includes two VMs 102a and 102b and consistency group 104b includes three VMs 103a, 103b, and 103c.

[0029] Each VM is associated with a set of resources that include information about the state of the VM. These resources include, for example, files associated with the VM such as log files, configuration files, and data files. The VMs may be associated with resources dedicated to that VM. For example, VM 103a is associated

with dedicated resources 126a, VM 103b is associated with dedicated resources 126b, and VM 103c is associated with dedicated resources 126c. The VMs may also be associated with linked resources. For example, VM 102a and VM 102b are both associated with a linked resource 124b (e.g., a linked file). These VMs 102a and 102b are also associated with dedicated resources 124a and 124c, respectively.

[0030] According to some embodiments, the invention is implemented by ensuring that the state of the resources for VMs within the same consistency group are captured and maintained on a consistent basis.

[0031] Fig. 2 shows a flowchart of an approach to capture the state of resources for consistency groups according to some embodiments of the invention. At 202, some or all of the VMs in the system are organized into consistency groups. As noted above, there may be multiple consistency groups in the system, where each consistency group may include any number of VMs.

[0032] Any suitable basis can be used to decide upon the members of a consistency group. As just one example, the consistency group can be organized to provide for data consistency across multiple VMs. This ensures, for example, that data dependencies that exist across multiple VMs do not turn into inconsistencies after a disaster recovery. As another example, there may be recognition that a set of multiple VMs pertain to closely related users, data, hardware, and/or subject matter, such that the VMs should be grouped together into a common consistency group.

[0033] A set of metadata is maintained in the system to track the membership of consistency groups. In some embodiments, the consistency group is structured as a

container object that includes the identifier of the VMs and/or VM resources that are mapped to the consistency group.

[0034] At 204, locks are acquired on the resources associated with the VMs of the consistency group. Lock management is a common approach that is used to synchronize accesses to shared resources. As noted above, the resource corresponds to any object pertaining to a VM to which shared access must be controlled. For example, the resource can be a file, a record, an area of shared memory, or anything else that can be shared by multiple VMs and/or entities in the system.

[0035] There are potentially many types of locks that may potentially be taken on the resource. In general, the lock should be of a type that precludes any modification to the resource while the system is in the midst of capturing the state of the resource. Examples of locks include, e.g., exclusive locks, protected read locks, and shared locks.

[0036] Once locks have been acquired on all of the appropriate resource, then at 206, a snapshot is taken of those resources. The snapshot is a recording of the state of the resource at a given moment in time. In effect, a synchronized snapshot is generated for every resource associated with the VMs in a consistency group that would be required in a disaster recovery situation to maintain the consistency across those multiple VMs in the same group.

[0037] The snapshot is then stored in a storage location that is predicted or anticipated to not share a common failure modality with the members of the consistency group. For example, a well-recognized failure mode in disaster scenarios is the failure of a power supply/source. Therefore, with recognition of this possible

failure scenario, it would make sense for the snapshot(s) of VMs for nodes attached to a first power supply/source to be stored in a storage location that is associated with a second power supply/source. Once the snapshots have been appropriated captured and are confirmed to be safely stored, the locks can then be released at 208. In some embodiments, the locks on the resource are released after the resource has been snapshotted, such as where the snapshotted resource is kept aside to perform fault-tolerance (e.g., replication) without blocking further writes to the resource.

[0038] Any suitable approach can be used to take a snapshot of a resource. For example, consider the situation when the application and/or the VM has either in-memory or on-disk state that needs to be snapshotted. The in-memory and/or on-disk data for the application/VM is stored in a set of one or more virtualized storage components. A copy of the data within the virtualized storage components is made and/or identified to perform the snapshot.

[0039] To explain, consider if the resources for the application/VM are stored as virtual disks or "vdisk", which is a logical representation of storage space compiled from one or more physical underlying storage devices. A file comprises data within one or more vdisks that are associated with the file. Metadata may be used to map the resources to the underlying physical storage devices. More information about an exemplary approach to implement vdisks and its associated metadata is described in co-pending U.S. Application Ser. Nos. 13/207,345 and 13/207,357, both filed on August 10, 2011, which are hereby incorporated by reference in their entirety.

[0040] When taking a snapshot, a copy is made and/or identified of the vdisks associated with a resource. Any suitable can be taken to make this type of copy of the

vdisks. In some embodiments, a copy-on-write approach is taken to make a copy of a vdisk when a change is made to that vdisk, where the previous version of the vdisk is still maintained. Both the previous and new version of the vdisk are associated with identifier numbers (e.g., "epoch" numbers) that can be used to distinguish between the different stored versions of the vdisks. For a given consistency group, snapshots for the vdisks associated with that consistency group would be taken at the same time, and therefore would be associated with the same epoch number.

[0041] In this way, any application can be snapshotted on a consistent basis, by implementing the application using virtualized storage (e.g., using the approach described in co-pending U.S. Application Ser. Nos. 13/207,345 and 13/207,357) and then snapshotting the virtual storage components associated with the application. This permits the consistency groups to be established in a way that is non-intrusive to the application/VM, and in which no special hooks are required in the VM/application to ensure that a collection of VMs can be snapshotted such that they are consistent.

[0042] This approach also permits any application, even one without a native capacity for snapshots, to implement consistency groups. To explain, consider an application that does not natively provide a capacity to implement snapshots, such as most modern non-database and/or non-disaster recovery applications. Most applications that are not themselves database management systems (DBMSs) or failure/disaster systems only offer rudimentary capabilities to handle data, without even the concept of point-in-time snapshots. With the present invention, the underlying storage for these applications is implemented using a virtualization system, e.g., where the application/application node is virtualized as a virtual machine and/or where the application uses a virtualized storage infrastructure having virtual

machines to manage its data. Using the above-described approach, consistent snapshots can then be taken of the data associated with the virtual machines that correspond to the application, even if the application code itself does not provide the ability to implement snapshots.

[0043] The actions of 204, 206, and 208 can be taken at any appropriate time periods, e.g., on a regular basis as established by a system administrator taking into account the needs to maintain up-to-date snapshots while balancing their costs. The snapshots can also be taken on an ad hoc basis at other time periods as well.

[0044] Fig. 3 shows a flowchart of an approach to restore the state of resources for VMs in a consistency groups after a disaster according to some embodiments of the invention.

[0045] At 302, identification is made of a consistency group that needs to be restored to implement disaster recovery. This may occur, for example, upon recognition of a disaster that has occurred which has brought down some or all of the VMs within a consistency group.

[0046] At 304, the appropriate snapshot(s) are identified for the consistency group to be restored. In some embodiments, the snapshots are stored within a hidden directory. The hidden directory is searched for the snapshot(s) of interest or the consistency group/VMs to be restored.

[0047] At 306, the identified snapshot(s) are associated with the VMs being restored to implement disaster recovery. For example, if the snapshots are stored in a hidden directory, then this step will move/copy the snapshots into a public namespace to be associated with the VMs that are being restored.

[0048] Thereafter, at 308, the VMs in the consistency group being restored are brought up using the data from the snapshot(s). Since the VMs are restored from snapshot(s) taken at a consistent point in time, this means that the VMs within the consistency group will be restored and brought up with an inherent consistency in their restored states.

[0049] Figs. 4A-E provide an illustrative example of the above-described approach to implement consistency groups. Fig. 4A shows a node 1 that is running VM 102a and VM 102b, both of which are members of the same consistency group 104a. These VMs 102a and 102b are associated with resources 124a, 124b, and 124c. As used herein, the term "node" refers to any appropriate computing entity and/or location, including without limitation, a machine, site, cluster, and/or system.

[0050] As illustrated in Fig. 4B, a snapshot 402 is taken of the resources 124a, 124b, and 124c. The snapshot 402 is taken to preserve a consistent state of 124a, 124b, and 124c at a specified moment in time.

[0051] Thereafter, as shown in Fig. 4C, a disaster occurs that results in failure of VMs 102a and 102b. Such a disaster may occur, for example, due to a hardware problem that takes down node 1. As a result, VMs 102a and 102b are no longer accessible to the user.

[0052] Disaster recovery is then pursued to bring VMs 102a and 102b back up. Since these two VMs are members for the same consistency group 104a, they must be restored in a manner that preserves the consistency of their restored states. It is assumed that the VMs will be restored using node 2 during the disaster recovery process.

[0053] Fig. 4D illustrates identification of snapshot 402 as the appropriate snapshot to implement the restoration of the VMs. The snapshot 402 is associated with a restored set of resources 124a-2, 124b-2, and 124c-2 for the restored VMs 102a-2 and 102b-2. As illustrated in Fig. 4E, when these VMs 102a-2 and 102b-2 are brought up, the state of the resources 124a-2, 124b-2, and 124c-2 that are accessed permit the VMs 102a-2 and 102b-2 to be restored to a consistent state from the time that the snapshot 402 was captured. This means that the VMs 102a-2 and 102b-2 within the restored consistency group 104a-2 has been restored and brought up with an inherent consistency in their restored states.

[0054] Therefore, what has been described above is an improved approach to implement disaster recovery for VMs, where consistency groups are provided to link together two or more VMs for disaster recovery purposes.

SYSTEM ARCHITECTURE

[0055] Fig. 5 is a block diagram of an illustrative computing system 1400 suitable for implementing an embodiment of the present invention. Computer system 1400 includes a bus 1406 or other communication mechanism for communicating information, which interconnects subsystems and devices, such as processor 1407, system memory 1408 (e.g., RAM), static storage device 1409 (e.g., ROM), disk drive 1410 (e.g., magnetic or optical), communication interface 1414 (e.g., modem or Ethernet card), display 1411 (e.g., CRT or LCD), input device 1412 (e.g., keyboard), and cursor control.

[0056] According to one embodiment of the invention, computer system 1400 performs specific operations by processor 1407 executing one or more sequences of one or more instructions contained in system memory 1408. Such instructions may be read into system memory 1408 from another computer readable/usable medium, such as static storage device 1409 or disk drive 1410. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and/or software. In one embodiment, the term "logic" shall mean any combination of software or hardware that is used to implement all or part of the invention.

[0057] The term "computer readable medium" or "computer usable medium" as used herein refers to any medium that participates in providing instructions to processor 1407 for execution. Such a medium may take many forms, including but not limited to, non-volatile media and volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as disk drive 1410. Volatile media includes dynamic memory, such as system memory 1408.

[0058] Common forms of computer readable media includes, for example, floppy disk, flexible disk, hard disk, magnetic tape, any other magnetic medium, CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, RAM, PROM, EPROM, FLASH-EPROM, any other memory chip or cartridge, or any other medium from which a computer can read.

[0059] In an embodiment of the invention, execution of the sequences of instructions to practice the invention is performed by a single computer system 1400.

According to other embodiments of the invention, two or more computer systems 1400 coupled by communication link 1415 (e.g., LAN, PTSN, or wireless network) may perform the sequence of instructions required to practice the invention in coordination with one another.

[0060] Computer system 1400 may transmit and receive messages, data, and instructions, including program, i.e., application code, through communication link 1415 and communication interface 1414. Received program code may be executed by processor 1407 as it is received, and/or stored in disk drive 1410, or other non-volatile storage for later execution. Data may be accessed/stored in a database 1432 on medium 1431 through a data interface 1433.

[0061] In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. For example, the above-described process flows are described with reference to a particular ordering of process actions. However, the ordering of many of the described process actions may be changed without affecting the scope or operation of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than restrictive sense.

Claims

What is claimed is:

1. A method for implementing a consistency group, comprising:

locking resources associated with the members of a consistency group, wherein the members of the consistency group comprises a group of related virtual machines;

generating a snapshot of the resources associated with the members of the consistency group; and

releasing locks on the resource after the snapshot has been generated.

2. The method of claim 1, in which an application associated with the resources does not have a native functionality to generate the snapshot.

3. The method of claim 1, in which an application associated with the resources is not modified to include functionality to implement the consistency group.

4. The method of claim 1, in which the snapshot comprises a copy of one or more virtual disks associated with the resources.

5. The method of claim 4, in which the snapshot comprises identification of one or more identifier numbers associated with the copy of the one or more virtual disks associated with the resources.

6. The method of claim 4, in which copy-on-write functionality is used to create the copy of the one or more virtual disks associated with the resources, wherein the one or more identifier numbers is incremented upon making the copy.

7. The method of claim 1, in which metadata is maintained to track membership in the consistency group.

8. The method of claim 1, in which the snapshot is generated at a time period using at least one of: a regular basis or an ad hoc basis.

9. The method of claim 1, further comprising:

identifying the consistency group to restore;

identifying the snapshot for the consistency group; and

initiating the group of related virtual machines using the snapshot.

10. The method of claim 9, in which the snapshot is associated with the group of related virtual machines.

11. The method of claim 1, in which disaster recovery is performed.

12. The method of claims 1-11 implemented as a system having means for implementing the method steps or as a computer program product comprising a computer-readable storage medium having executable code to execute the method steps.

13. A method for restoring a consistency group, comprising:

identifying a consistency group to restore to implement disaster recovery, wherein the consistency group comprises a group of related virtual machines;

identifying a snapshot associated with the consistency group; and

using the snapshot to bring up the virtual machines in the consistency group in a consistent manner.

14. The method of claim 13 implemented as a system having means for implementing the method steps or as a computer program product comprising a computer-readable storage medium having executable code to execute the method steps.

FIG. 1

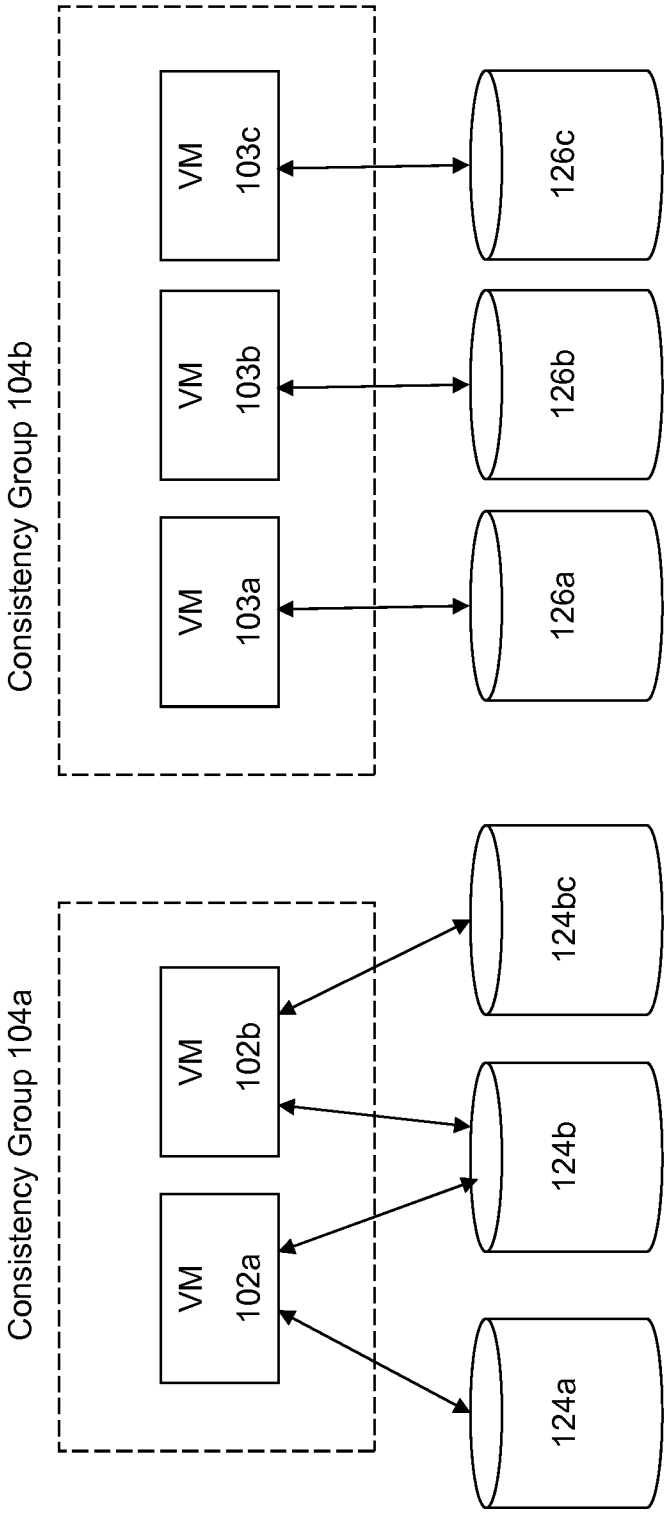

Consistency Group 104b
VM 103c
VM 103b
VM 103a
126c
126b
126a
Consistency Group 104a
VM 102b
VM 102a
124bc
124b
124a

FIG. 2

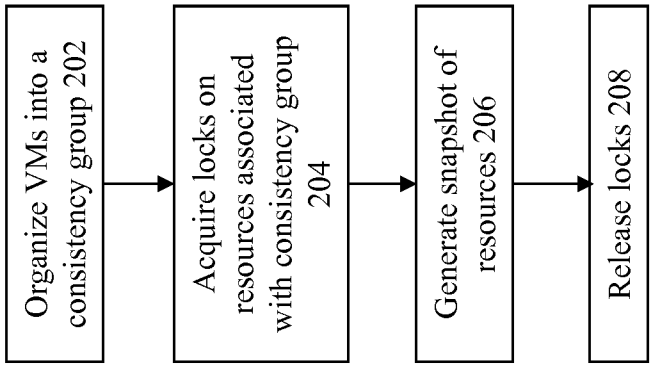

Organize VMs into a consistency group 202
Acquire locks on resources associated with consistency group 204
Generate snapshot of resources 206
Release locks 208

FIG. 3

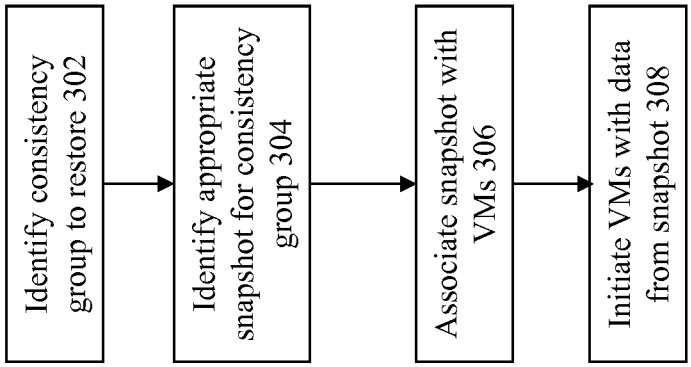

Identify consistency group to restore 302
Identify appropriate snapshot for consistency group 304
Associate snapshot with VMs 306
Initiate VMs with data from snapshot 308

FIG. 4A

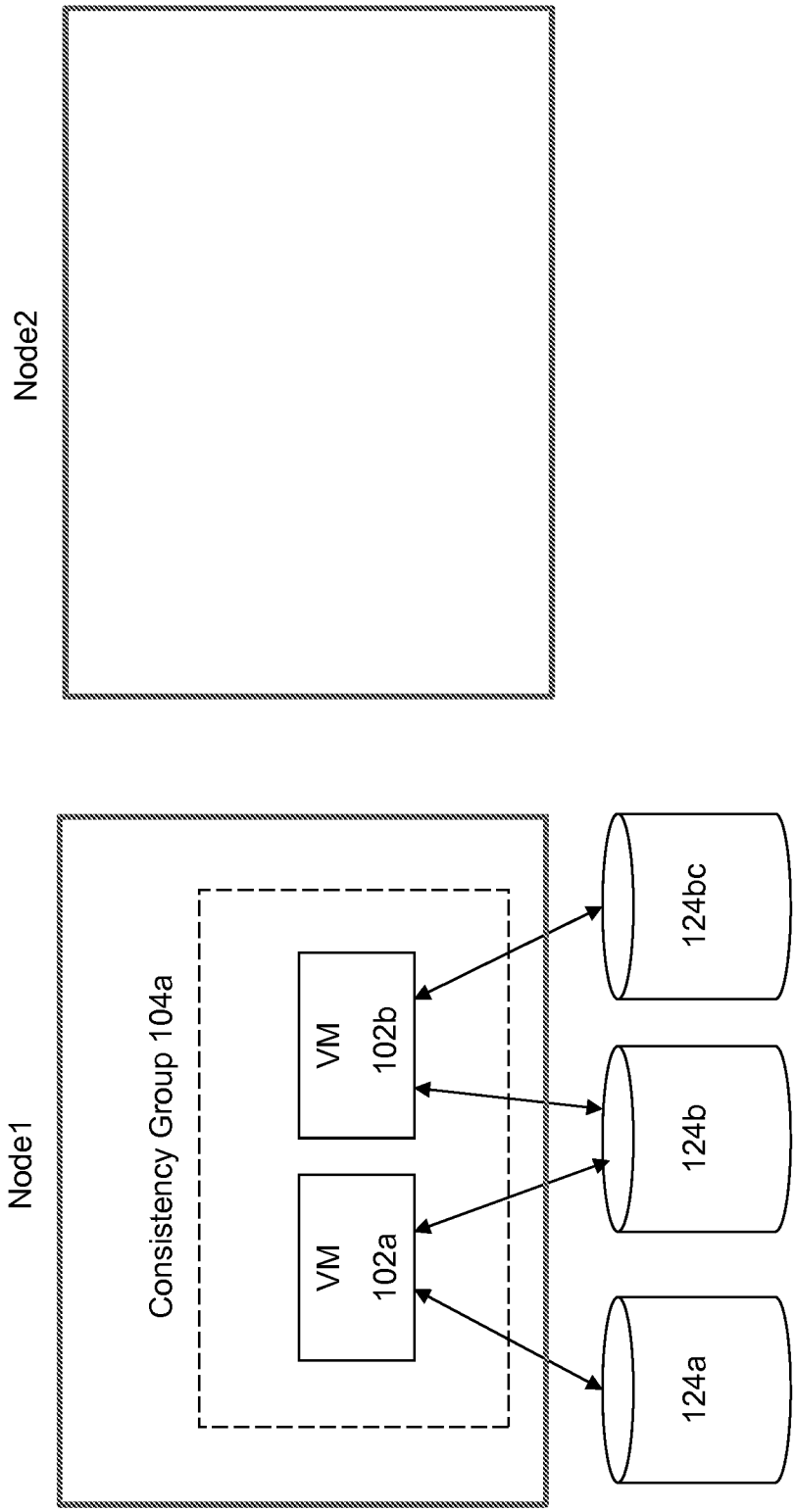

Node2
Node1
Consistency Group 104a
VM
102b
VM
102a
124bc
124b
124a

FIG. 4B

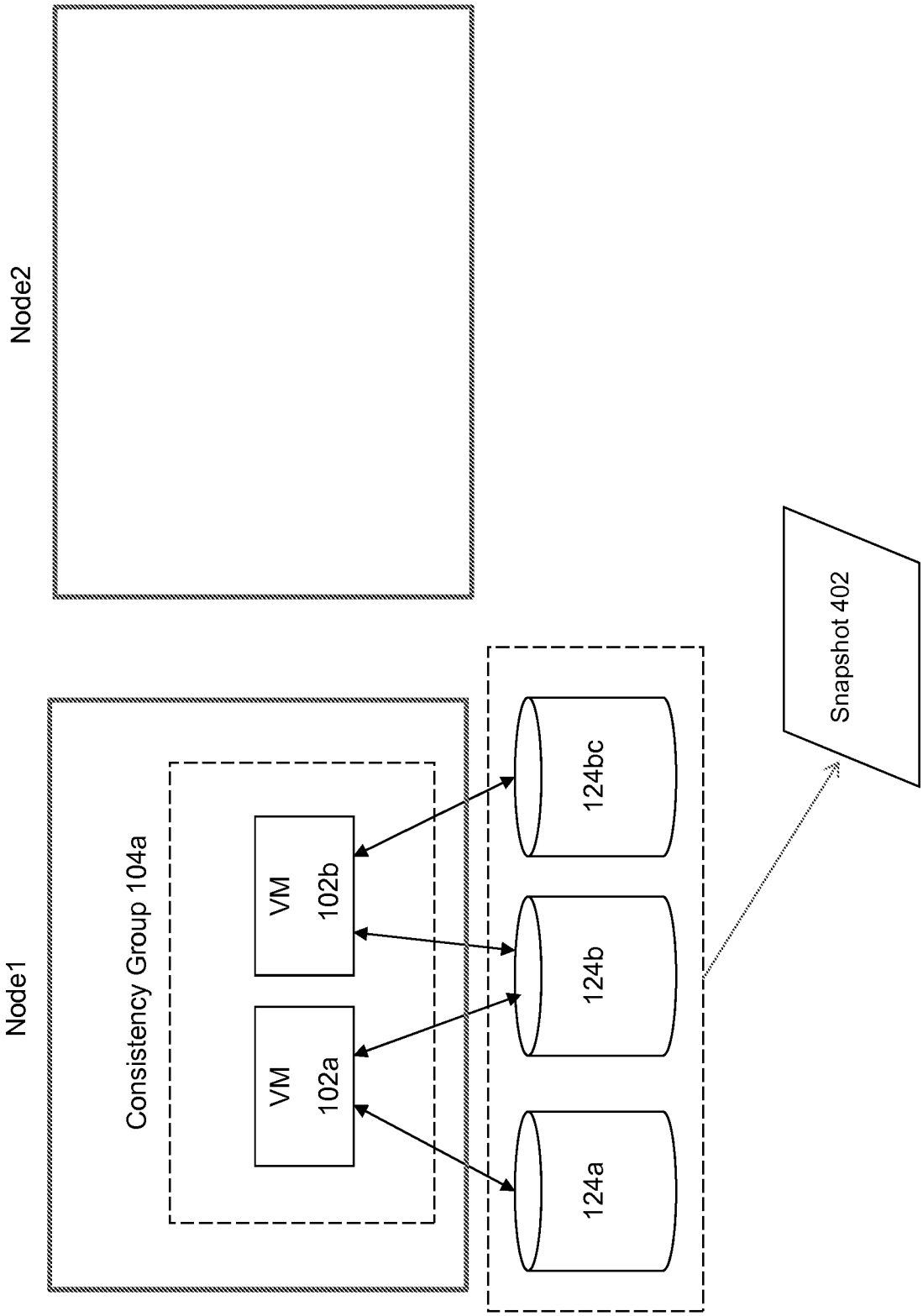

Node2
Node1
Consistency Group 104a
VM 102b
VM 102a
124bc
124b
124a
Snapshot 402

FIG. 4C

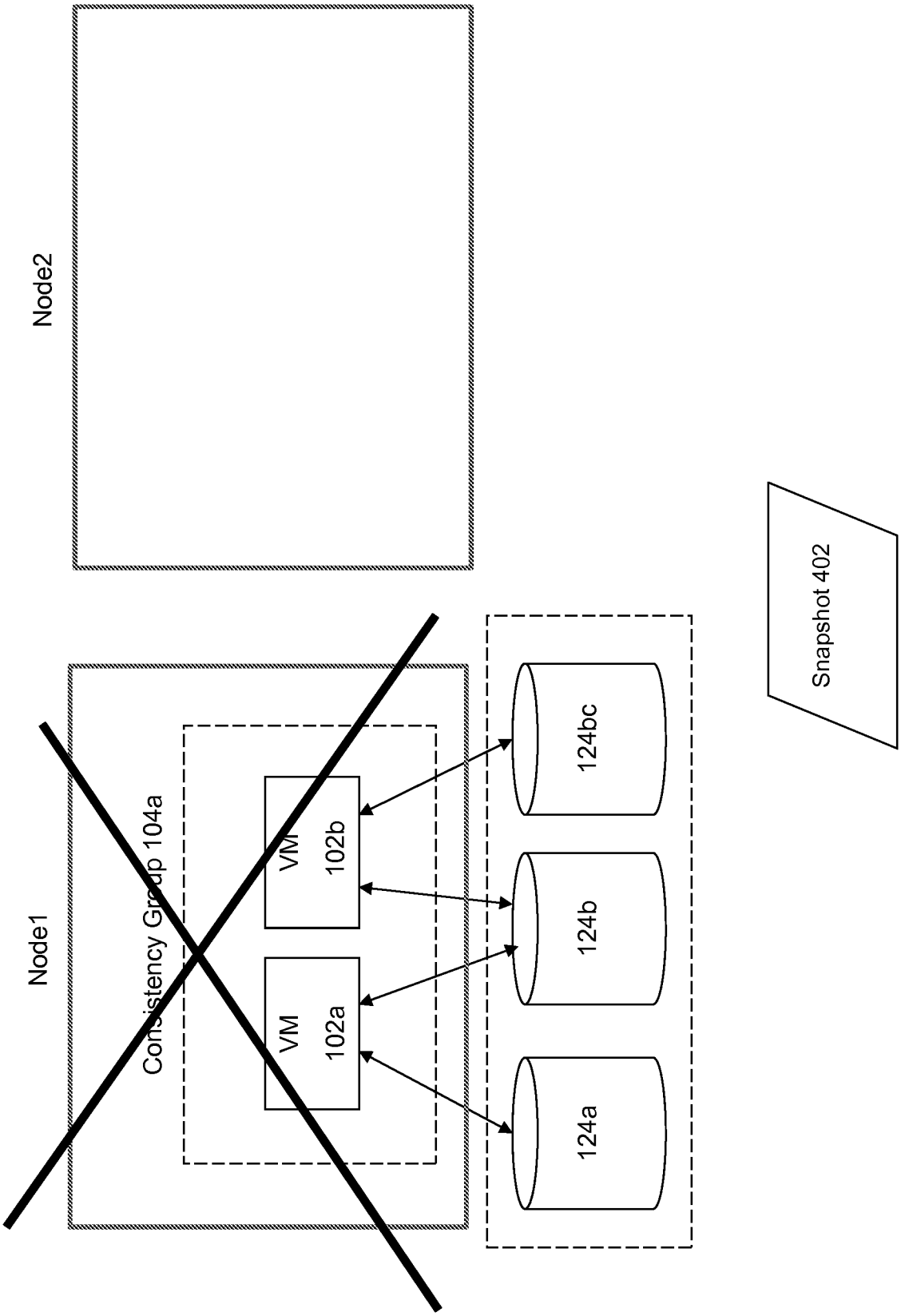

Node2
Node1
Consistency Group 104a
VM
102b
VM
102a
124bc
124b
124a
Snapshot 402

FIG. 4D

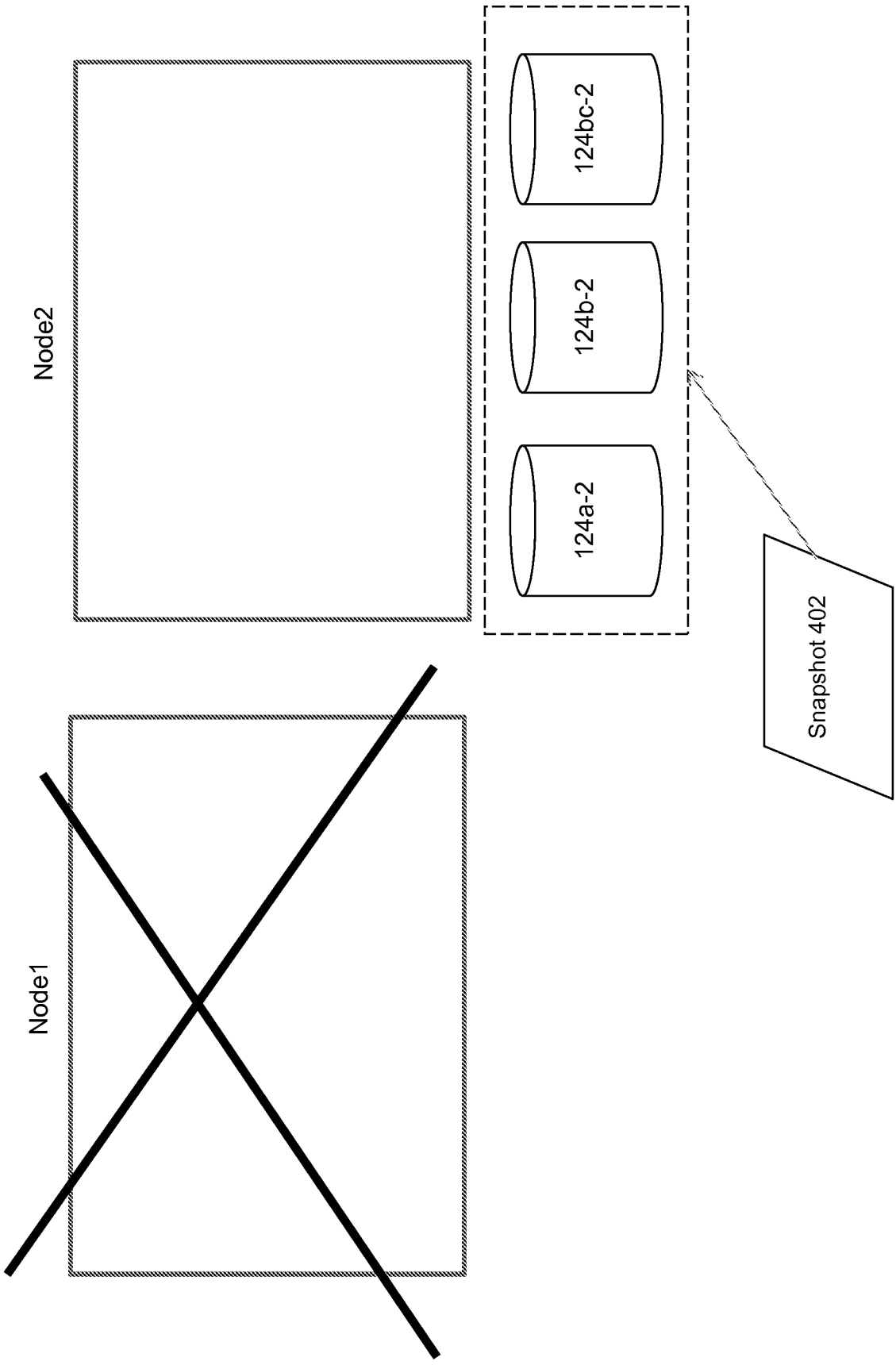


FIG. 4E

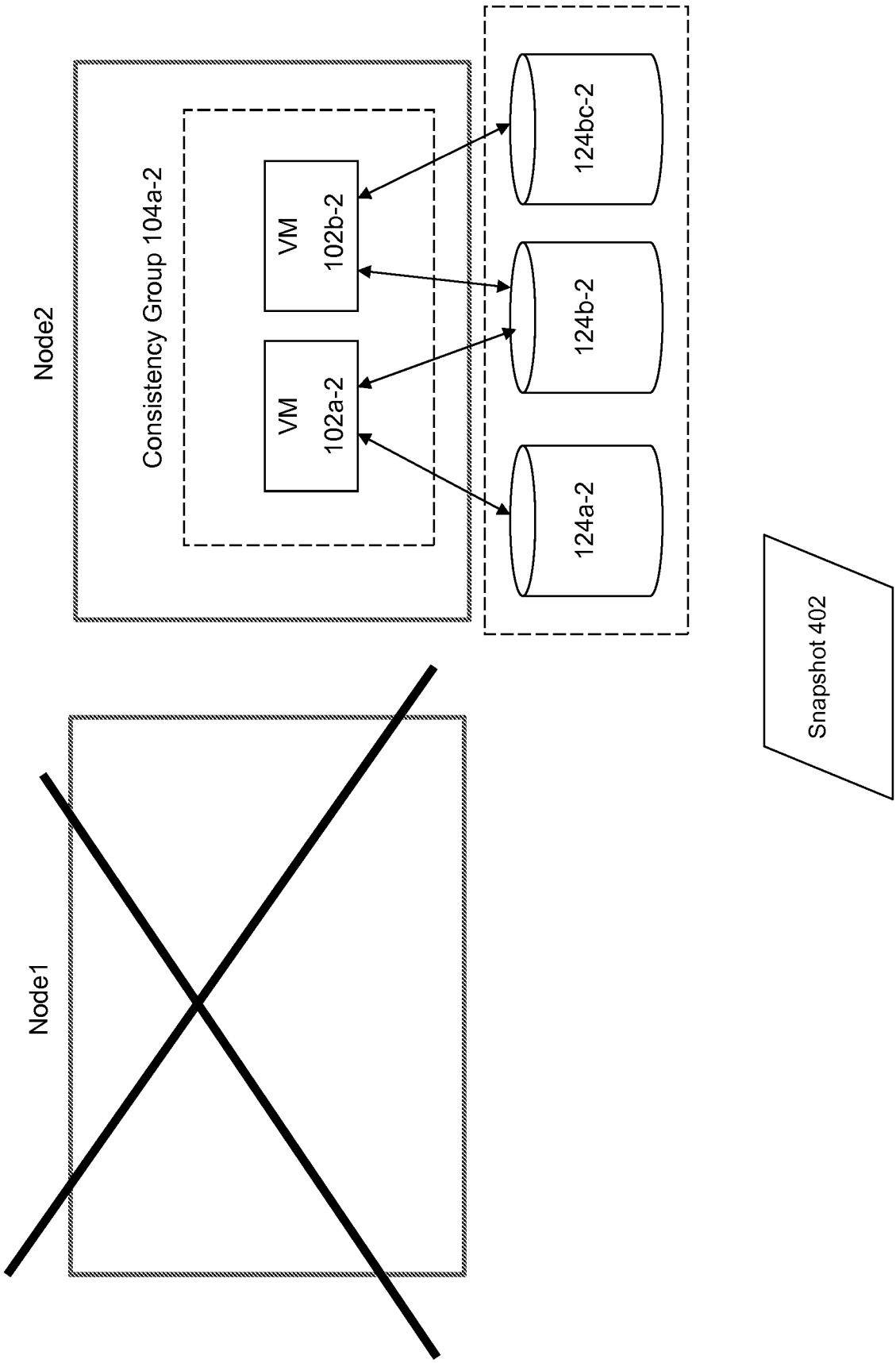
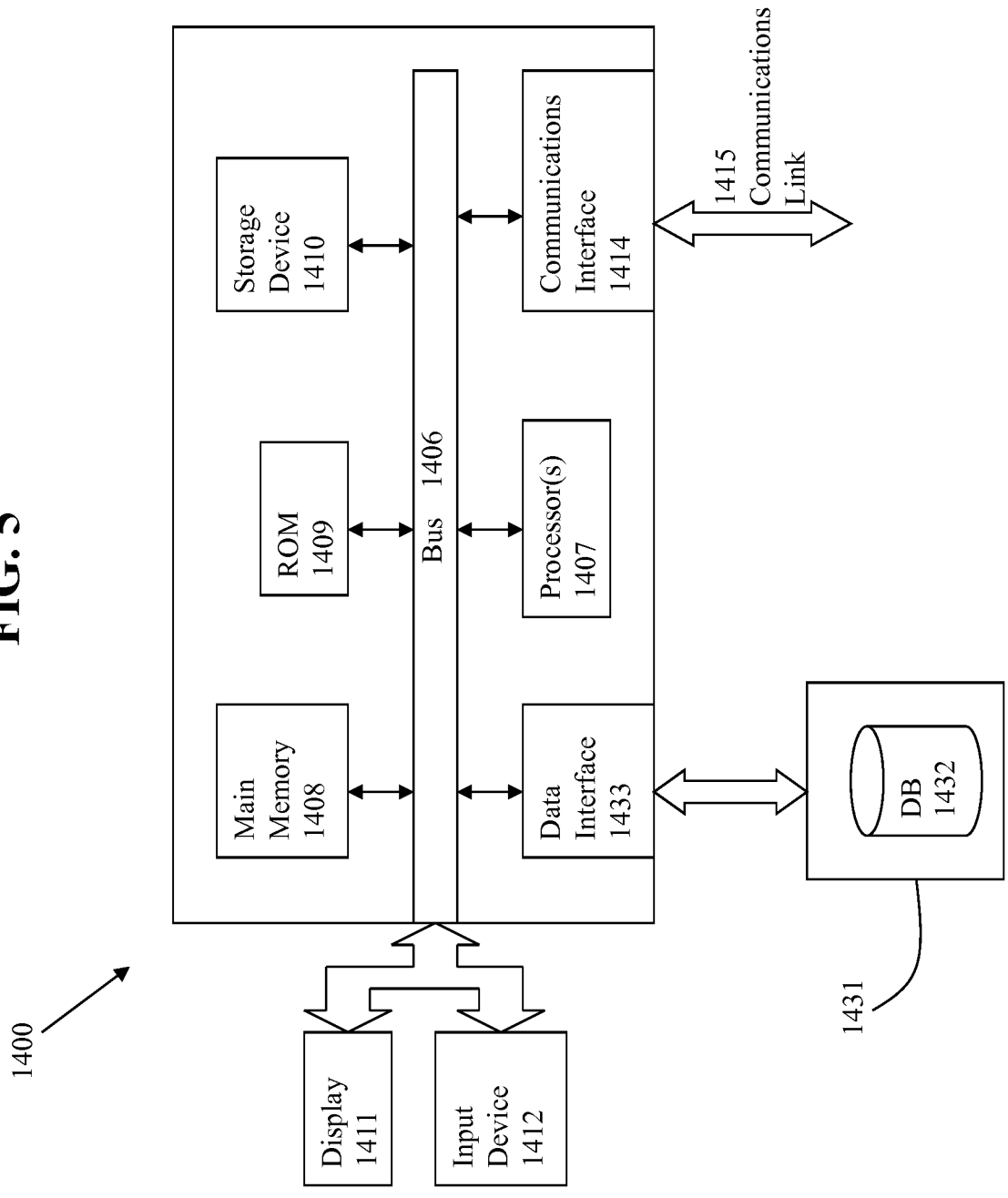

Node2
Consistency Group 104a-2
VM 102a-2
VM 102b-2
124a-2
124b-2
124bc-2
Node1
Snapshot 402

FIG. 5


1400
Display 1411
Input Device 1412
Main Memory 1408
ROM 1409
Storage Device 1410
Bus 1406
Processor(s) 1407
Data Interface 1433
Communications Interface 1414
1415 Communications Link
1431
DB 1432