(12)Indian Patent Application

---

(21) Application Number: 202341063878

(22) Filing Date: 22/09/2023 (43) Publication Date: 28/03/2025

(71) Applicant(s): L&T TECHNOLOGY SERVICES LIMITED

(72) Inventor(s): Singha, Zubraj

(51) International Classifications: G06F 9/455 G06F 3/06 G06F 16/28 H04N 19/46 H01L 21/02

(54) Title: METHOD AND SYSTEM OF MANAGING ACCESS OF DANGLING PERSISTENT VOLUMES

(57) Abstract: A system (100) and method (200) of managing access of one or more persistent volumes (122) by a hypervisor (106) is disclosed. The hypervisor (106) determines an I/O request from an I/O queue corresponding to at least one of the persistent volumes (122). A dangling flag is determined corresponding to the at least one of the persistent volumes (122) by the hypervisor (106). Upon determining the dangling flag as TRUE, the hypervisor (106) moves the data corresponding to the at least one of the persistent volumes (122) to a buffer.

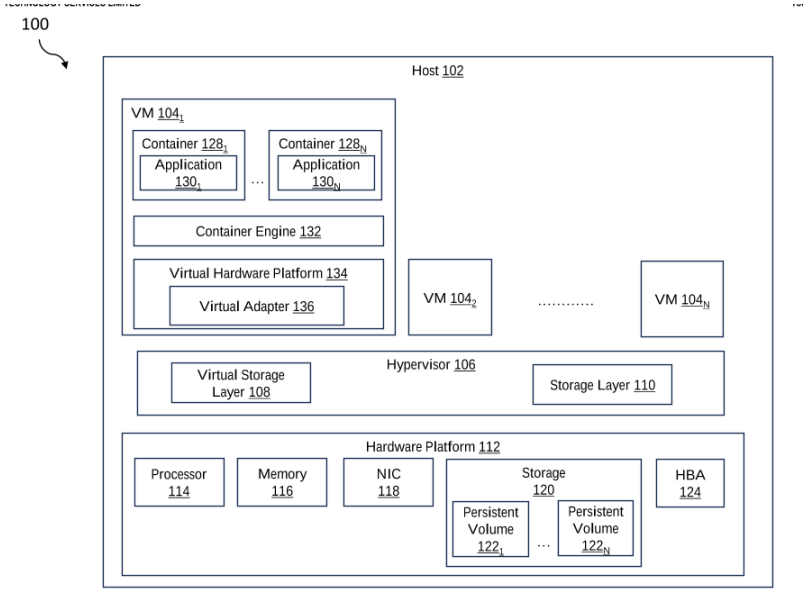

FIG. 1

FORM 2

THE PATENTS ACT 1970
(39 OF 1970)
&
The Patent Rules, 2003

Complete Specification
(See Section 10 and Rule 13)

1. TITLE OF THE INVENTION

METHOD AND SYSTEM OF MANAGING ACCESS OF DANGLING PERSISTENT VOLUMES

2. APPLICANT(S)

(a) NAME : **L&T TECHNOLOGY SERVICES LIMITED**
(b) NATIONALITY : **INDIAN**
(c) ADDRESS : DLF IT SEZ Park, 2nd Floor – Block 3
1/124, Mount Poonamallee Road,
Ramapuram, Chennai – 600 089,
INDIA.

3. PREAMBLE TO THE DESCRIPTION

COMPLETE

The following specification particularly describes the invention and the manner in which it is to be performed

DESCRIPTION

Technical Field

[001] This disclosure relates generally to a virtual environment in a server, more particularly to a method and system of managing access of persistent volumes.

BACKGROUND

[002] In order to preserve data generated by the running container, and also to share data between containers, persistent volumes are created in a storage of a host where the container is running. However, the lifecycle of the persistent volume does not depend on the lifecycle of the container, thereby the persistent volume persists even after the container is deleted. The purpose of a persistent volume is to exist independent from its container, therefore, when container is removed, the persistent volume may be referred to as dangling persistent volume which may be left dangling without any user access. In general, the user has to manually delete these persistent volumes once the containers are deleted but due to human error or lack of diligence the persistent volumes may not be deleted after the containers are deleted. This may allow such dangling persistent volumes susceptible to malicious attacks. Accordingly, any critical data saved in such dangling persistent volume is exposed without supervision.

Also, due to no monitoring and access of dangling persistent volume from any containers, there is no way to validate the data for corruption.

[003] Thus, there is a need to provide a methodology of managing access of persistent volumes.

SUMMARY OF THE INVENTION

[004] In an embodiment, a method of managing access of one or more persistent volumes is disclosed. The method may include determining, by a hypervisor of a host, an I/O request from an I/O queue corresponding to at least one of the persistent volumes. The method may further include determining a dangling flag corresponding to the at least one of the persistent volumes. In an embodiment, the dangling flag may be set based on an association of the at least one of the persistent volumes with a corresponding container in the host. The method may further include moving data corresponding to the at least one of the persistent volumes to a buffer volume upon determining the dangling flag as true.

[005] In another embodiment, a system of managing access of one or more persistent volumes is disclosed. The system may include a processor and a memory coupled to the processor. The memory may store processor-executable instructions which when executed may cause the processor to determine an I/O request from an I/O queue corresponding to at least one of the persistent volumes. The processor may further determine a dangling flag corresponding to the at least one of the persistent volumes. In an embodiment, the dangling flag may be set based on an association of the at least one of the persistent volumes with a corresponding container in the host. The processor may further move data corresponding to the at least one of the persistent volumes to a buffer volume upon determining the dangling flag as true.

[006] Various objects, features, aspects, and advantages of the inventive subject matter will become more apparent from the following detailed description of preferred embodiments, along with the accompanying drawing figures in which like numerals represent like components.

BRIEF DESCRIPTION OF THE DRAWINGS

[007] The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, serve to explain the disclosed principles.

[008] **FIG. 1** depicts an architecture diagram of an access management system for managing access of persistent volumes, in accordance with an embodiment of the current disclosure.

[009] **FIG. 2** illustrates a flowchart describing the methodology for managing access of persistent volumes, in accordance with an embodiment of the current disclosure.

DETAILED DESCRIPTION OF THE DRAWINGS

[010] Exemplary embodiments are described with reference to the accompanying drawings. Wherever convenient, the same reference numbers are used throughout the drawings to refer to the same or like parts. While examples and features of disclosed principles are described herein, modifications, adaptations, and other implementations are possible without departing from the scope of the disclosed embodiments. It is intended that the following detailed description be considered as exemplary only, with the true scope being indicated by the following claims. Additional illustrative embodiments are listed.

[011] Further, the phrases “in some embodiments”, “in accordance with some embodiments”, “in the embodiments shown”, “in other embodiments”, and the like mean a particular feature, structure, or characteristic following the phrase is included in at least one embodiment of the present disclosure and may be included in more than one embodiment. In addition, such phrases do not necessarily refer to the same embodiments or different embodiments. It is intended that the following detailed description be considered exemplary only, with the true scope being indicated by the following claims.

[012] In case of deletion of a container, there may be no access to its associated persistent volume which should ideally be deleted. However, such persistent volume when not deleted may be referred to as dangling persistent volume and there may be no way to validate the data on the dangling persistent volume for corruption. The present disclosure provides a methodology for managing access of persistent volumes in a better manner.

[013] Referring now to **FIG. 1,** an architecture diagram of an access management system 100 for managing access of persistent volumes in a virtual computing environment in a host 102, in accordance with an embodiment of the current disclosure. As shown in **FIG. 1**, the access management system 100 may include one or more hosts 102 which may be enabled on a server grade hardware such as an x86 architecture platform. The access management system 100 may enable a hypervisor 106 that may be enabled on a server (not shown) and may run on top of an OS in the host 102. In some implementations, hypervisor 106 may comprise system level software as well as a “Domain 0” or “Root Partition” VM, which is a privileged machine that has access to the hardware platform 112 resources of host 102

[014] In an embodiment, the hypervisor 106 may abstract the processor 114, memory 116, storage 120 and one or more networking resources in a hardware platform 112.

[015] The hardware platform 112 may also include a network interface card (NIC) 118 including one or more network adapters, an HBA 124, and other I/O devices. Further, the hypervisor 106 may be communicably coupled via network (not shown) to one or more virtual machines 104_1, 104_2, …,104_n (individually referred to as VM 104 and collectively referred to as VMs 104) that may run concurrently on the same host 102. The network can be implemented as one of the different types of networks, such as but not limited to, ethernet IP network, intranet, local area network (LAN), wide area network (WAN), the internet, Wi-Fi, LTE network, CDMA network, 5G and the like. Further, network can either be a dedicated network

or a shared network. The shared network represents an association of the different types of networks that use a variety of protocols, for example, Hypertext Transfer Protocol (HTTP), Transmission Control Protocol/Internet Protocol (TCP/IP), Wireless Application Protocol (WAP), and the like, to communicate with one another. Further network can include a variety of network devices, including routers, bridges, servers, computing devices, storage devices, and the like. The server may be implemented on one or more computer servers.

[016] In an embodiment, examples of the processor(s) 114 may include, but are not limited to, an Intel® Itanium® or Itanium 2 processor(s), or AMD® Opteron® or Athlon MP® processor(s), Motorola® lines of processors, Nvidia®, FortiSOC™ system on a chip processors or other future processors. The processor 114 may be configured to execute instructions, for example, executable instructions that perform one or more operations described herein and that may be stored in memory 116 and in storage 120.

[017] The memory 116 may be hardware allowing information, such as executable instructions, configurations, and other data, to be stored and retrieved. The memory 116 may be where programs and data may be kept when processor 114 may be actively using them. The memory 116 may be volatile memory or non-volatile memory. Volatile or non-persistent memory may need constant power in order to prevent data from being erased. Volatile memory may describe conventional memory, such as dynamic random-access memory. Non-volatile memory may be memory that may be persistent (non-volatile). Non-volatile memory may be byte-addressable random access non-volatile memory. Examples of non-volatile memory may include but are not limited to, a flash memory, a Read Only Memory (ROM), a Programmable ROM (PROM), Erasable PROM (EPROM), and Electrically EPROM (EEPROM) memory. Further, examples of volatile memory may include but are not limited to, Dynamic Random Access Memory (DRAM), and Static Random-Access memory (SRAM).

[018] The NIC 118 may enable host 102 to communicate with other devices via a communication medium. The HBA 124 may couple the host 102 to one or more external storages (not shown), such as a storage area network (SAN). Other external storage that may be used may include network-attached storage (NAS) and other network data storage systems, which may be accessible via NIC 118.

[019] The storage 120 may represent persistent storage devices. In an embodiment, examples of persistent storage devices may be one or more hard disks, flash memory modules, solid state

drives (SSDs), and/or optical disks). Although the exemplary embodiment shown in **FIG. 1** illustrates storage 120 as local storage in hardware platform 112. In some other embodiments, storage 120 may be directly coupled to the host 102. In an embodiment, the storage 120 may be a virtual storage area network (VSAN) that aggregates local or direct-attached capacity devices of a host cluster, including the host 102, and creates a single storage pool shared across hosts in the cluster.

[020] The hypervisor 106 and the one or more virtual machines 104 and a virtual file system layer (not shown) may be stored in the memory 116 for use by a server operating system (not shown). Further, one or more programs may operate directly on a virtual hardware platform 134.

[021] In an embodiment, the virtual file system layer may interface with virtual hardware platform 134 to access a data storage host bus adapter (HBA) 124 that may be implemented by the virtual hardware platform 134 to provide the appearance of disk storage support to enable execution of VMs 104, a virtual adapter 136 implemented by a host bus adapter 124 in the hardware platform 124. Accordingly, the virtual hardware platform 134 may provide the appearance of disk storage support to enable execution of the VMs 104 based on the virtualization of the hardware platform 112. In an embodiment, a virtual disk may expose the same abstraction as a physical disk, in form of, but not limited to, a linear list of sectors. However, the data transfer and control operations may be passed through various layers of hypervisor 106 to true hardware HBAs 124 or network interface cards (NICs) 118 that may connect to the storage 120.

[022] In an embodiment, the storage 120 may include one or more persistence volumes $122_{1\text{-}N}$ (individually referred to as persistent volume 122 and collectively referred to as persistent volumes 122) each corresponding to one or more containers $128_{1\text{-}N}$ (individually referred to as container 128 and collectively referred to as containers 128). Accordingly, each of the persistent volumes 122 may be storage for each of the corresponding containers 128 that may be provisioned by an administrator, manually, dynamically or automatically. In an embodiment, each of the persistent volumes 122 may represent a physical disk or a file system local to the host 102.

[023] In an embodiment, the VMs 104 may include a container engine 132 that may be installed and run as a guest application under control of the VMs 104. The container engine

132 may be a process that enables the deployment and management of the one or more containers 128 by providing a layer of OS-level virtualization within VMs 104. The containers 128 may be software instances that may enable virtualization at the OS level with containerization. From the standpoint of an end user that may communicate with each of the containers 128 may appear to be communicating to unique servers. However, from the standpoint of the OS of the host 102 on which the containers 128 may execute, the containers 128 may appear as user processes that may be scheduled and dispatched by the OS. In an embodiment, example of the container engine 132 may include, but not limited to, the open-source Docker platform.

[024] In an embodiment, each of the containers 128 may encapsulate one or more applications $130_{1...}130_{N}$ (collectively referred to as applications 130 and individually referred to as application 130). In an embodiment, the applications 130 may behave as a single executable package of software that bundles application code together with all of the related configuration files, libraries, and dependencies required for it to run. The applications 130 may be any software program, such as, but not limited to, a word processing program, etc.

[025] In an embodiment, each of the persistent volumes 122 may be a plugin volume having lifecycle independent of the lifecycle of its corresponding container 128. Accordingly, when the container 128 associated with a persistent volume 122 may be removed from the host 102 (e.g., the container 128 is no longer running or deleted), the persistent volume 122 may still remain in the storage 120. Accordingly, the persistent volume 122 may be referred to as a dangling volume indicating the persistent volume 122 may not be associated to any of the running containers 128.

[026] In an embodiment, one layer of the hypervisor 106 may be but not limited to a virtual virtual storage layer 108 that, for example, may receive commands from the virtual adapter 136, in the form of SCSI command blocks, the virtual storage layer 108 may further convert the SCSI command blocks into the file system operations that may be understood by a storage layer 110. The virtual storage layer 108 may verify the identity of a container 128 requesting access to a persistent volume 122 reserved by the hypervisor 106 to prevent unauthorized parties from accessing the persistent volume 122. The storage layer 110 may be configured to manage storage space for VMs 104. In an embodiment, the hypervisor 106 may enable a background thread daemon for each of the persistent volume 122. The background thread

daemon may read the I/O requests in the virtual adaptor queue or I/O request queue and may be awakened from a sleep state based on detection of a conditional variable.

[027] In an exemplary scenario, at least one of the persistent volumes 122 may become a dangling persistent volume when a container 128 associated with the at least one persistent volume 122 is killed (e.g. terminated or deleted) by an administrator of the container 128. In such a case, the at least one persistent volume 122 may no longer be needed and the administrator may also remove the dangling persistent volume 122 which was associated with killed container 128. For the period of time between when container 128 is killed and the at least one persistent volume 122 is removed, data maintained for the killed container 128 in the persistent volume 122 may be vulnerable to attack. In particular, an attacker who may have gained unauthorized access to the VM 104 in the user space of the host 102 may subsequently access sensitive data maintained by the at least one persistent volume 122.

In some other cases, at least one persistent volumes 122 may become a dangling volume when the container 128 associated with the persistent volume 122 may crash. In such a case, the persistent volume 122 associated with container 128 which has crashed may be retained for use by a new container that may be generated from a container image of the container 128 which previously crashed (e.g., container 128 is a live, running instance of a container image). When a new container is spawned by an administrator, the new container may reclaim ownership of the dangling persistent volume 122. However, for the period of time between when container 128 crashes and a new container is spawned for reclamation of the dangling persistent volume 122, data maintained for the container 128 in the dangling persistent volume 122 may be vulnerable to attack.

[028] By way of an example, the memory 116 may store processor-executable instructions, which, on execution, causes the processor 114 to enable the hypervisor 106 to manage access of one or more persistent volumes 128. In an embodiment, the hypervisor 106 may be configured to determine an I/O request from an I/O queue via the virtual adaptor 136. The hypervisor 106 may enable a dangling flag in each of the persistent volumes 122 created for access by each of the containers 128. It is to be noted that the dangling flag status at initiation may be set as "FALSE" by the hypervisor 106.

[029] Further, a background thread daemon of each of the persistent volumes 122 may be in a sleep mode and may be configured to monitor a conditional variable. In an embodiment, the

conditional variable may be a dangling flag of each of the persistent volumes 122. In an exemplary embodiment, the conditional variable of a persistent volume 122 may be defined as, but not limited to, “isDangling”.

[030] In case a persistent volume 122 is accessed through an I/O request, the background thread daemon of the persistent volume 122 may monitor the dangling flag of the persistent volume 122. Further, in case the dangling flag of the persistent volume 122 is determined as “TRUE” the background thread daemon may be awakened or activates to signal to the hypervisor 106 to take an action in order to secure the data of the persistent volume 122. In an embodiment, the dangling flag of at least one of the persistent volumes 122 may be set as “TRUE” in case the corresponding container 128 of the at least one of the persistent volumes 122 has been deleted or has crashed, thus converting such persistent volume 122 to a dangling persistent volume. In an exemplary embodiment, the dangling flag of such a dangling persistent volume 122 may be set as “TRUE” by the hypervisor 106 in case of deletion or crash of the corresponding container 128. Accordingly, the dangling flag of any of the persistent volumes 122 may be set based on detection of an association of the persistent volume 122 with their corresponding containers 128 in the host 102. Accordingly, the dangling flags in each of the persistent volumes 122 may be initiated as “FALSE” and may be set as “TRUE” based on detection an association of the persistent volumes 122 with their corresponding containers 128.

[031] In an embodiment, the dangling flag of a dangling persistent volume 122 may be determined as “TRUE” by issuing an IOCTL from the host 102. In an embodiment, the IOCTL may be defined as, but not limited to, “Persistent_Disk_Dangling”. Accordingly, upon determining the dangling flag as “TRUE”, the background thread daemon may be activated by the hypervisor 106. In an embodiment, the background thread daemon upon being awakened or activated may signal the hypervisor 106 to take an action to secure the data of the persistent volumes 122. In an embodiment, the hypervisor 106 may in order to secure the data of the dangling persistent volume 122 may move the data stored on such dangling persistent volume 122 to a buffer volume upon determining the dangling flag as “TRUE”. In an embodiment, the buffer volume may be a secured ringbuffer volume. In an embodiment, the ringbuffer volume may be reserved on each of the one or more persistent disks 122 that may store a list of commands which are sent when a persistent disk 122 is in a dangling state or has its dangling flag set as “TRUE”. In another embodiment, upon determining the dangling flag as “TRUE” for a persistent volume 122 and upon receiving an I/O request for the corresponding dangling

persistent volume 122, the hypervisor 106 may display an alert on a display screen to prompt an administrator to monitor the buffer volume of the corresponding dangling persistent volume 122 for an unauthorized access or corruption. Further, in another embodiment, upon receiving an I/O request for the corresponding dangling persistent volume 122, the hypervisor 106 may display an alert on a display screen to prompt an administrator to delete the dangling persistent volume 122. In another embodiment, the hypervisor 106 may monitor a time period for which the dangling flag of the dangling persistent volume 122 is set as "TRUE" and may delete the dangling persistent volume 122 after a predefined time period has passed. Further, the hypervisor 106 may display an alert on a display screen to prompt an administrator to delete the dangling persistent volume 122 after a predefined time period has passed since the dangling flag is set as "TRUE".

[032] Referring now to **FIG. 2,** a flowchart describing the methodology for managing access of persistent volumes, in accordance with an embodiment of the current disclosure. In an embodiment, method 200 may include a plurality of steps that may be performed by hypervisor 106 to manage access of persistent volumes 122.

[033] At step 202, an I/O request may be determined from an I/O queue corresponding to at least one of the persistent volumes 122. At step 204, the hypervisor 106 may determine if a dangling flag corresponding to the at least one of the persistent volumes 122 is set as "TRUE". In an embodiment, the dangling flag corresponding to the at least one of the persistent volumes 122 may be set based on association of the at least one of the persistent volumes 122 with a corresponding container 128 in the host 102. In general, a dangling flag corresponding to each of the one or more persistent volumes 122 may be initialized as "FALSE".

[034] Further, in case the dangling flag of the at least one of the persistent volumes 122 is determined as "TRUE", the hypervisor 106 may move the data corresponding to the at least one of the persistent volumes 122 to a buffer volume.

[035] In an embodiment, the dangling flag of the at least one of the persistent volumes 122 may be determined as "TRUE" when the corresponding container may be deleted or may have crashed in the host 102. Further, in an embodiment, the dangling flag of the at least one of the persistent volumes 122 may be determined as "TRUE" by issuing an IOCTL from the host 102.

[036] Further, in case at step 206, the dangling flag of the at least one of the persistent volumes 122 may be determined as "FALSE", the hypervisor 106 may continue to check the dangling

flag of the at least one of the persistent volumes 122 for which a subsequent I/O request may be received in the I/O queue at step 204.

[037] It is intended that the disclosure and examples be considered as exemplary only, with a true scope of disclosed embodiments being indicated by the following claims.

WE CLAIM:

1. A method (200) of managing access of one or more persistent volumes (122), the method (200) comprising:

determining (202), by a hypervisor (106) of a host (102) from an I/O queue, an I/O request corresponding to at least one of the persistent volumes (122);

determining (204), by the hypervisor (106), a dangling flag corresponding to the at least one of the persistent volumes (122),

wherein the dangling flag is set based on an association of the at least one of the persistent volumes with a corresponding container in the host; and

upon determining the dangling flag as TRUE:

moving (206), by the hypervisor (106), data corresponding to the at least one of the persistent volumes to a buffer volume.

2. The method (200) as claimed in claim 1, wherein the dangling flag is set as TRUE when the corresponding container (128) is deleted in the host (102).

3. The method (200) as claimed in claim 2, wherein the dangling flag is determined as TRUE by issuing an IOCTL from the host (102).

4. The method (200) as claimed in claim 1, wherein upon determining the dangling flag as TRUE, a background thread daemon is activated by the hypervisor (106).

5. The method (200) as claimed in claim 1, wherein a dangling flag corresponding to each of the one or more persistent volumes (122) is initialized as FALSE.

6. A system (100) of managing access of one or more persistent volumes (122) in a host (102), the system (100) comprises:

a processor (114); and

a memory (116) coupled to the processor (114), wherein the memory (116) stores processor-executable instructions, which, on execution, causes the processor (114) to enable a hypervisor (106), wherein the hypervisor (106) is configured to:

determine from an I/O queue, an I/O request to least one of the persistent volumes (122);

determine a dangling flag corresponding to the at least one of the persistent volumes (122),

wherein the dangling flag is set based on an association of the at least one of the persistent volumes (122) with a corresponding container (128) in the host (102); and

upon determining the dangling flag as TRUE:

move data corresponding to the at least one of the persistent volumes (122) to a buffer volume.

7. The system (100) as claimed in claim 6, wherein the dangling flag is set as TRUE when the corresponding container (128) is deleted in the host (102).

8. The system (100) as claimed in claim 7, wherein the dangling flag is set as TRUE by issuing an IOCTL from the host (102).

9. The system (100) as claimed in claim 6, wherein upon determining the dangling flag as TRUE, the hypervisor (106) is configured to activate a background thread daemon.

10. The system (100) as claimed in claim 6, wherein the hypervisor (106) is configured to initialize a dangling flag corresponding to each of the one or more persistent volumes (122) as FALSE.

Dated this 22nd day of September 2023

--Digitally Signed--
Bhanu Prasad (INPA No: 3253)
Head, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

ABSTRACT

METHOD AND SYSTEM OF MANAGING ACCESS OF DANGLING PERSISTENT VOLUMES

A system (100) and method (200) of managing access of one or more persistent volumes (122) by a hypervisor (106) is disclosed. The hypervisor (106) determines an I/O request from an I/O queue corresponding to at least one of the persistent volumes (122). A dangling flag is determined corresponding to the at least one of the persistent volumes (122) by the hypervisor (106). Upon determining the dangling flag as TRUE, the hypervisor (106) moves the data corresponding to the at least one of the persistent volumes (122) to a buffer.

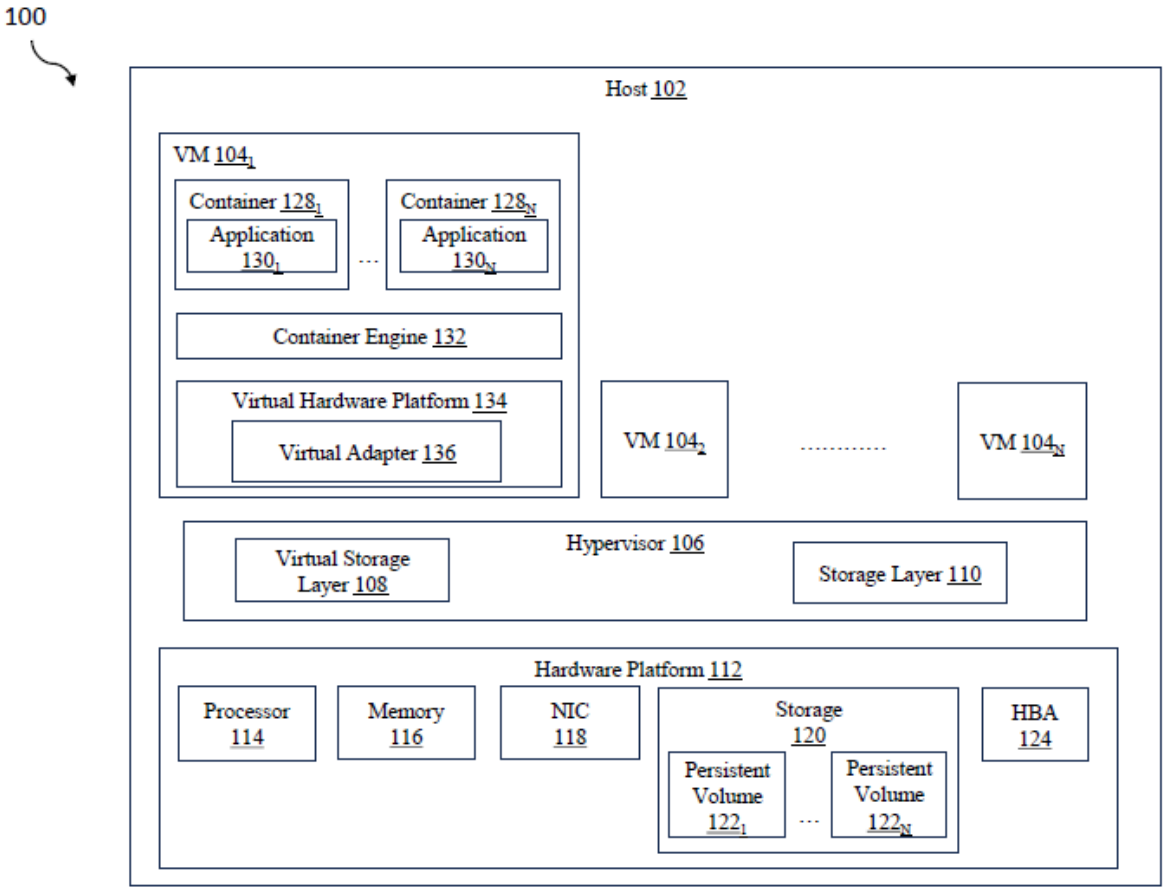

FIG. 1

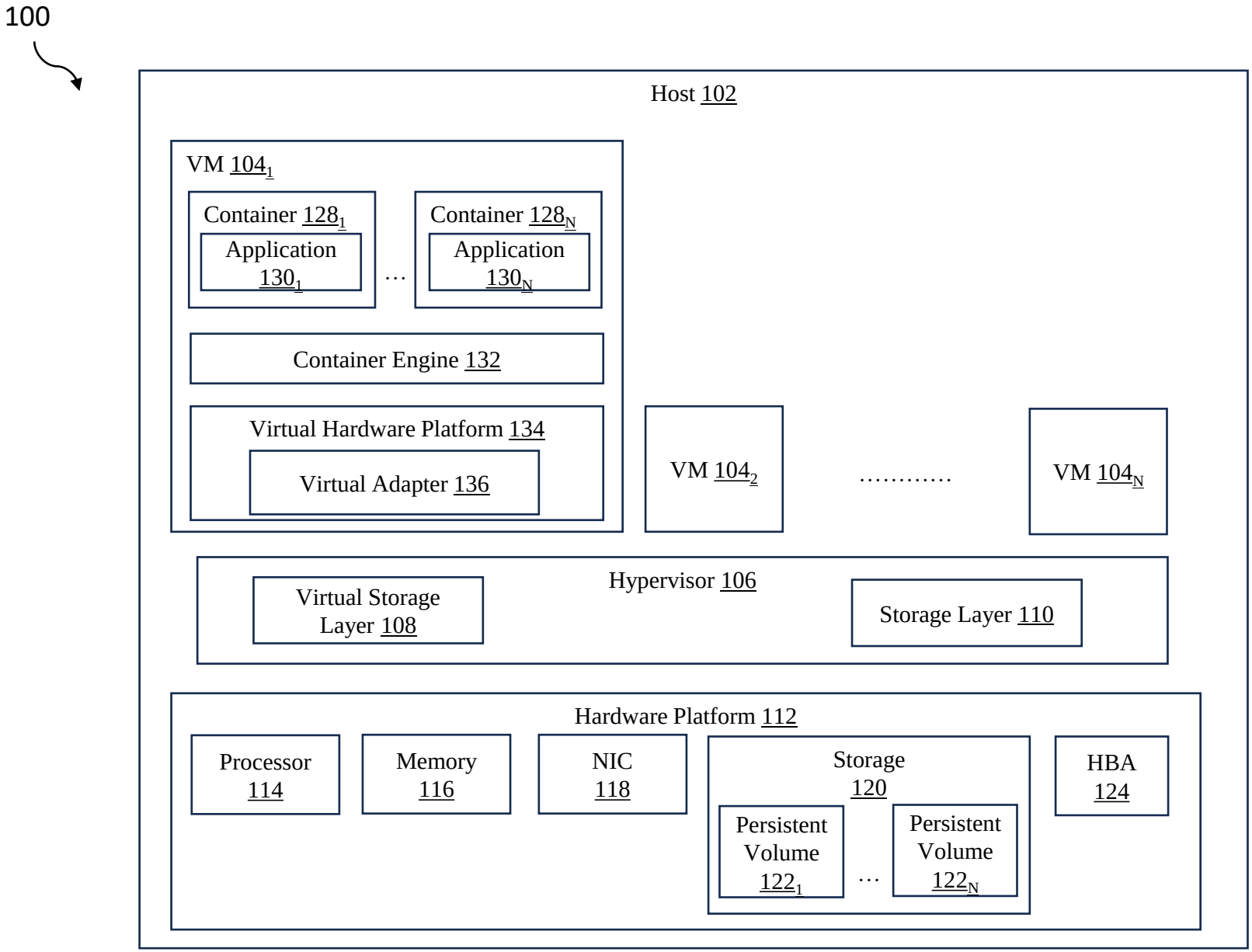

FIG. 1

200

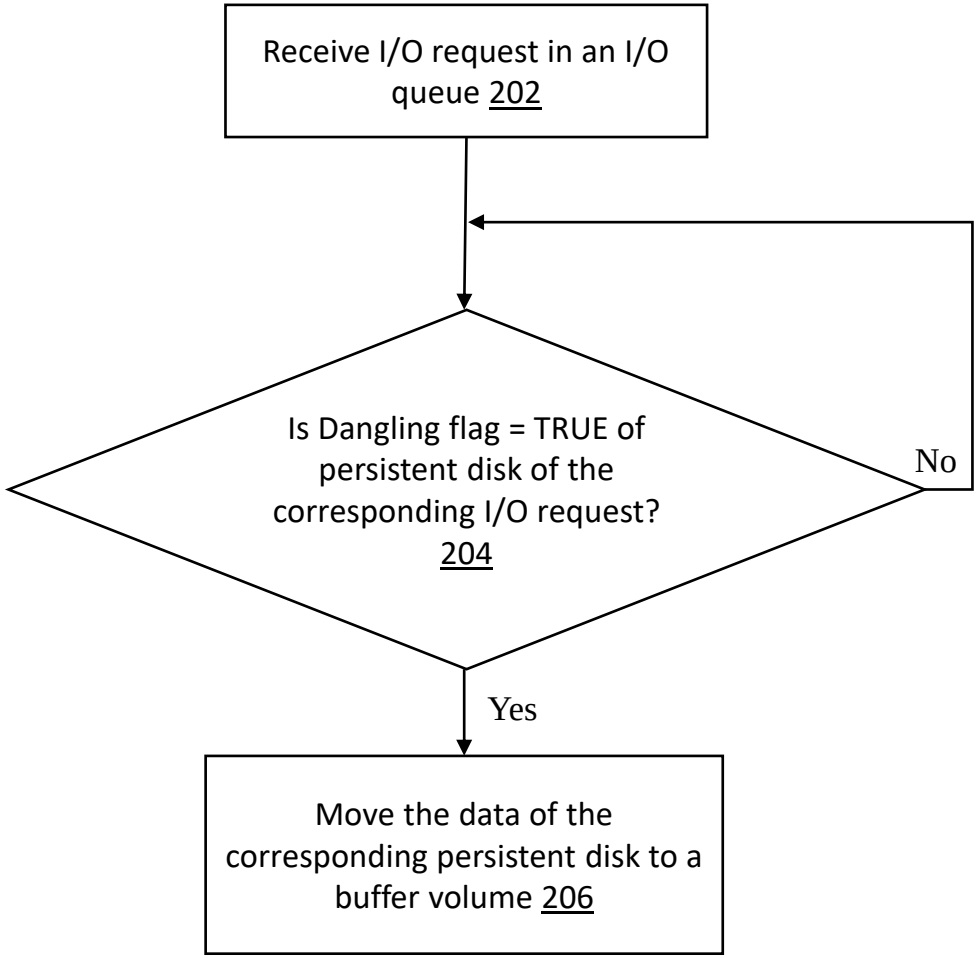

FIG. 2