(12)Indian Patent Application

---

(21) Application Number: 202341088402

(22) Filing Date: 22/12/2023 (43) Publication Date: 27/06/2025

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

(72) Inventor(s): Singha, Zubraj

(51) International Classifications: G06F 9/455 G11B 27/034 G06F 11/14 H01S 3/06 G11B 7/00

(54) Title: METHOD AND SYSTEM OF SECURING DATA OF AT LEAST ONE PERSISTENT DISK IN A HYPERVISOR ENVIRONMENT

(57) Abstract: A system (100) and method (300) of securing data of at least one persistent disk (122) in a hypervisor environment is disclosed. A hypervisor (106) determines a status of the at least one persistent disk (122) based on a dangling flag corresponding to the at least one persistent disk (122). Upon determining the dangling flag as TRUE, the hypervisor (106) provisions a thin disk (124) of volume about same or greater than a volume of the at least one persistent disk (122). Further, the hypervisor (106) reads data corresponding to each of a plurality of sectors of the at least one persistent disk (122) in a linear manner by a background thread. Further, the hypervisor (106) writes the linearly read data onto the thin disk (124) based on a pseudorandom algorithm.

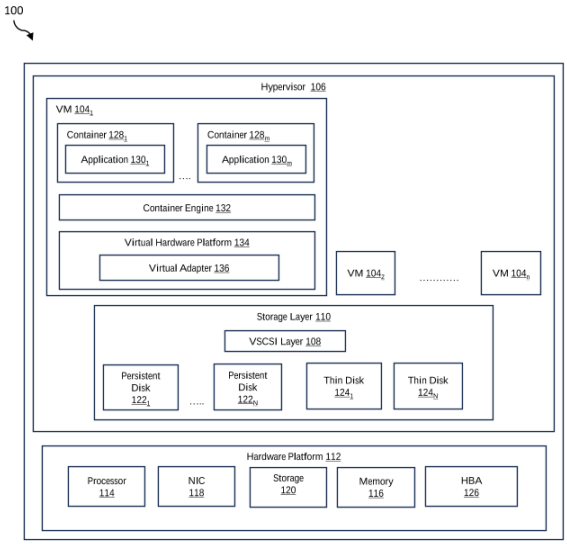

FIG.1

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 SECURING DATA OF PERSISTENT DISKS

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 securing data of persistent disks in a hypervisor environment.

BACKGROUND

[002] To secure data generated by a running container, and also to share data between containers, persistent disks are created in the storage of a host where the containers are running. However, the lifecycle of the persistent disk does not depend on the lifecycle of the container, thereby a persistent disk may persist even after all the containers are deleted. However, a persistent disk that exists independent from any container, such persistent disk may be referred to as a dangling persistent disk. However, such dangling persistent disks may include critical data that may be susceptible to malicious attacks. In order to secure data of such dangling persistent disks, a user should manually delete such persistent disks once the containers are deleted. However, due to human error or lack of diligence, the dangling persistent disks may not be deleted, thus making them prone to malicious data attacks.

[003] Thus, there is a need to provide a methodology for securing data of dangling persistent disks in a hypervisor environment.

SUMMARY OF THE INVENTION

[004] In an embodiment, a method of securing data of at least one persistent disk in a hypervisor environment is disclosed. The method may include determining, by a hypervisor, a status of the at least one persistent disk. In an embodiment, the status of the at least one persistent disk may be determined as dangling based on a dangling flag associated to the at least one persistent disk. In an embodiment, the dangling flag may be set based on an association of the at least one persistent disk with corresponding one or more containers in the hypervisor environment. Upon determining the dangling flag of the at least one persistent disk as TRUE, the method may further include, provisioning, by the hypervisor, a thin disk of volume about same or greater than a volume of the at least one persistent disk. Further, the method may include, reading, by the hypervisor, data corresponding to each of sectors of the at least one persistent disk in a linear manner by a background thread. The method may further

include, writing, by the hypervisor, the linearly read data onto the thin disk based on a pseudo-random algorithm.

[005] In another embodiment, a system of securing data of at least one persistent disk in a hypervisor environment is disclosed. The system may include a processor and a memory coupled to the processor, wherein the memory may store processor-executable instructions, which when executed by the processor may cause the processor to enable a hypervisor, wherein the hypervisor is configured to determine a status of the at least one persistent disk. In an embodiment, the status of the at least one persistent disk may be determined as dangling based on a dangling flag associated to the at least one persistent disk. In an embodiment, the dangling flag may be set based on an association of the at least one persistent disk with corresponding one or more containers in the hypervisor environment. Upon determining the dangling flag of the at least one persistent disk as TRUE, the hypervisor may provision a thin disk of volume about same or greater than a volume of the at least one persistent disk. Further, the hypervisor may read data corresponding to each of the sectors of the at least one persistent disk in a linear manner by a background thread. Further, the hypervisor may write the linearly read data onto the thin disk based on a pseudo-random algorithm.

[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 a hypervisor environment, in accordance with an embodiment of the present disclosure.

[009] **FIG. 2A** illustrates an exemplary diagram of a persistent disk storing data at various sector locations, in accordance with an embodiment of the present disclosure.

[010] **FIG. 2B** illustrates an exemplary diagram of a thin disk storing data at various sector locations, in accordance with an embodiment of the present disclosure.

[011] **FIG. 2C** illustrates an exemplary diagram of a dangling persistent disk storing data transferred from the thin disk, in accordance with an embodiment of the present disclosure.

[012] **FIG. 3** illustrates a flowchart of a method for securing data of the persistent disk, in accordance with an embodiment of the present disclosure.

DETAILED DESCRIPTION OF THE DRAWINGS

[013] 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.

[014] 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.

[015] In case of deletion or crash of a container, its associated persistent disk may be rendered dangling as there is no direct user access to such dangling persistent disk. In an ideal condition, such a disk should be deleted. However, when left undeleted such dangling persistent disks are often left unattended and become susceptible to malicious attacks. Further, there may be no way to validate the data on the dangling persistent disk for corruption.

[016] In computing environments, data is often written and stored linearly. This linear storage structure means that data is stored in a continuous and sequential fashion, where each piece of information is laid out one after another. For instance, let's consider a virtual disk in a

hypervisor environment. Data within this virtual disk is stored in a linear fashion, where each block of data is arranged one after the other. When the system writes new data, it does so in a sequential manner, ensuring that the information follows a specific order. Therefore, an attacker aiming to access this data must comprehend and understand the order and the underlying logic used to write the data. In case the data is not written linearly, the attacker would need to understand how the data is written and organized. Else, the data even if read by the attacker would be incomprehensible. The present disclosure provides a methodology for securing data of at least one persistent disk in an efficient manner.

[017] Referring now to **FIG. 1,** an architecture diagram of a hypervisor environment 100, in accordance with an embodiment of the current disclosure. As shown in **FIG. 1**, the hypervisor environment 100 may include one or more hosts (not shown) that may be enabled on a server grade hardware such as an x86 architecture platform. The hypervisor environment 100 (also referred to as virtualized environment) 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 (not shown). In some implementations, the 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 a hardware platform 112 resources of the host (not shown). In an embodiment, the hypervisor 106 may abstract the processor 114, memory 116, storage 120 and one or more networking resources in the hardware platform 112.

[018] The hardware platform 112 may also include a network interface card (NIC) 118 including one or more network adapters, an HBA 126, and other I/O devices. Further, the hypervisor 106 may be communicably coupled via a 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 (not shown). 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.

[019] 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.

[020] 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).

[021] The NIC 118 may enable host to communicate with other devices via a communication medium. The HBA 126 may couple the host 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.

[022] 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. 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, and creates a single storage pool shared across hosts in the cluster.

[023] 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.

[024] In an embodiment, the virtual file system layer may interface with virtual hardware platform 134 to access a data storage host bus adapter (HBA) 126 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 112. 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 126 or network interface cards (NICs) 118 that may connect to the storage 120.

[025] In an embodiment, the storage layer 110 may include one or more persistent disks 122_{1-N} (individually referred to as persistent disk 122 and collectively referred to as persistent disks 122) each corresponding to one or more containers 128_{1-N} (individually referred to as container 128 and collectively referred to as containers 128). Accordingly, each of the persistent disks 122 may be storage for each of the corresponding one or more containers 128 that may be provisioned by an administrator, manually, dynamically or automatically. In an embodiment, each of the persistent disks 122 may represent a physical disk or a file system local to the host. In an embodiment, the storage layer 110 may further include one or more thin disks 124_{1-N} (individually referred to as thin disk 124 and collectively referred to as thin disks 124) each corresponding to the one or more persistent disks 122_{1-N}. In an embodiment, volume of the thin disks 124 may be about same or greater than a volume of the corresponding persistent disks 122_{1-N}. In an embodiment, each of the thin disks 124 may represent a physical disk or a file system local to the host (not shown).

[026] 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, communicating with each of the containers 128, it may appear to be communicating to unique servers. However, from the standpoint of the OS of the host 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.

[027] 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.

[028] In an embodiment, each of the persistent disks 122 may be a plugin disk having lifecycle independent of the lifecycle of its corresponding one or more containers 128. Accordingly, when the one or more containers 128 associated with a persistent disk 122 may become non-operational in the host in case the one or more containers 128 are deleted or have crashed or become corrupt. In such a scenario, the persistent disk 122 associated to such non-operational one or more containers 128 may still persist in the storage layer 110. In an embodiment, such persistent disk may be referred to as a dangling persistent disk indicating that the persistent disk 122 may not be associated to any of the running containers 128.

[029] In an embodiment, the hypervisor 106 may enable a background daemon for each of the persistent disk 122. A background thread may read the I/O requests in the virtual adaptor queue or I/O request queue. In an exemplary scenario, at least one of the persistent disks 122 may become a dangling persistent disk when one or more containers 128 associated with the at least one persistent disk 122 is non-operational (e.g. crashed or deleted) by an administrator of the container 128. In a scenario, when the administrator would want to restore the non-operational one or more containers 128, the data of the corresponding persistent disks 122 may

be exposed to malicious attack for a period of time between when the one or more containers 128 become non-operational and the time when the one or more containers 128 are restored. In particular, an attacker who may have gained unauthorized access to the VM 104 in the user space of the host may subsequently access sensitive data maintained by such dangling persistent disk 122.

[030] In some other embodiment, at least one persistent disk 122 may become a dangling volume when the container 128 associated with the persistent disk 122 may crash. In such a case, the persistent disk 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 disk 122, data saved in the dangling persistent disk 122 may be vulnerable to attack.

[031] 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 secure data of at least one persistent disk 122. 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 disks 122 created for access by each of the containers 128 as explained in detail in previously filed application titled “METHOD AND SYSTEM OF MANAGING ACCESS OF DANGLING PERSISTENT VOLUMES” and having application number IN202341063878, incorporated herein by reference in its entirety. It is to be noted that the dangling flag status at initiation may be set as "FALSE” for each persistent disks 122 by the hypervisor 106.

[032] Further, a background thread of each of the persistent disks 122 may be in a sleep mode and may be configured to determine status of each of the persistent disks 122. In an embodiment, the status of a persistent disk 122 may be determined as dangling in case the dangling flag of the corresponding persistent disk 122 is set as “TRUE” based on determination of an association of the corresponding persistent disk 122 with one or more corresponding containers 128 in the hypervisor environment 100. In an embodiment, a persistent disk 122 having a dangling flag set as “TRUE” may be referred to as a dangling persistent disk 122.

[033] In case a persistent disk 122 is accessed through an I/O request, the background thread daemon of the persistent disk 122 may monitor the dangling flag of the persistent disk 122. Further, in case the dangling flag of the persistent disk 122 is determined as "TRUE" the background thread daemon may be awakened or activated to signal to the hypervisor 106 to take an action in order to secure the data of the dangling persistent disks 122. In an embodiment, the dangling flag of at least one of the persistent disks 122 may be set as "TRUE" in case the associated one or more containers 128 of the at least one of the persistent disk 122 become non-operational. Accordingly, the dangling flag of any of the persistent disks 122 may be set based on detection of an association of the persistent disks 122 with their corresponding one or more containers 128 in the hypervisor environment 100. Accordingly, the dangling flags in each of the persistent disks 122 may be initiated as "FALSE" and may be set as "TRUE" based on detection an association of the persistent disks 122 with their corresponding containers 128.

[034] Upon determining the dangling flag as "TRUE", the background thread may be activated by the hypervisor 106. In an embodiment, the background thread upon being awakened or activated may signal the hypervisor 106 to secure the data of the persistent disks 122 that has been determined as having a dangling status.

[035] Referring now to **FIG. 2A,** a persistent disk 122 having data stored in various sectors locations is illustrated, in accordance with an embodiment of the present disclosure is illustrated. In an exemplary embodiment, the persistent disk 122 may store exemplary data depicted using annotations (X0, X1, X2, X3, X4, X5) at exemplary sector locations depicted using annotations (S0, S1, S2, S3, S4, S5) respectively of the persistent disk 122. As will be appreciated, the exemplary data has been depicted using annotations (X0, X1, X2, X3, X4, X5) for representation purpose, however, such annotations are representative of respective data values having a predefined size based on sector size of the persistent disk 122. Further, the sector locations of the persistent disk 122 have been depicted using annotations (S0, S1, S2, S3, S4, S5) respectively, however, such annotations are representative of a sequence of sectors present in the persistent disk 122. In an exemplary embodiment, a sector size may be, but not limited to, 512 bytes. In an exemplary embodiment, a number of sectors in a persistent disk 122 of size 10Gb may be calculated based on formula given by equation (1) given below:

$$\text{Number of Sectors} = ((10 * 1024*1024*1024)/(\text{sector size})) \quad \ldots\ldots (1)$$

[036] In case of a sector size of 512 bytes, the number of sectors in a persistent disk may be 2,09,71,520. Referring back to **FIG. 1,** the hypervisor 106 may, in order to secure the data (X0, X1, X2, X3, X4, X5) stored in the dangling persistent disk 122 may block input/output access to the dangling persistent disk 122 until the background thread may complete the reading of all the data (X0, X1, X2, X3, X4, X5) and writing of the linearly read data (X0, X1, X2, X3, X4, X5) onto a newly provisioned thin disk 124.

[037] Further, the hypervisor 106 may in order to secure the data of the dangling persistent disk 122 may provision the thin disk 124 of volume about same or greater than a volume of the corresponding persistent disk 122. According to the exemplary embodiment of **FIG. 2A**, the volume of thin disc 124 provisioned may be equal to or greater than 10 MB.

[038] Accordingly, the hypervisor 106 may further enable the background thread to read data in each of the plurality of sectors (S0, S1, S2, S3, S4, S5) of the at least one persistent disk 122 in a linear manner. The hypervisor 106 may further enable the background thread to write each of the linearly read data onto the thin disk 124 based on a pseudo-random algorithm. In an embodiment, the hypervisor 106 may use a random number generated based on the pseudo-random algorithm as an offset sector location of the thin disk 124. Accordingly, in an exemplary embodiment, the linearly read data from each of the sectors (S0, S1, S2, S3, S4, S5) may be written onto the thin disk by the background thread based on the linearly generated random number based on the pseudo-random algorithm.

[039] In an embodiment, each of the random numbers generated may be a non-repeated number for a range and a seed of the pseudo-random algorithm. In an embodiment, the seed may be an initial value used to initialize the random number generator. Further, a range of the pseudo-random algorithm may also be pre-defined as a span of numbers within which the generated random numbers may fall. In another embodiment, in case a random number generated based on the pseudo-random number is same as a previously occurred random number then the repeated random number may be skipped, and a next subsequent non-repetitive random number may be used as the offset sector location on the thin disk 124.

[040] In an embodiment, the range and the seed of the pseudo-random algorithm may be determined based on a sector size of the dangling persistent disk 122 and a word length of the data saved in each of the plurality of sectors of the at least one persistent disk 122. In an exemplary embodiment, the seed of the pseudo-random algorithm may be determined by

reading ten linear offsets from the sectors and doing a bitwise XOR of every word saved on the ten sectors. For instance, sector size * 10 = 5120 bytes. In an embodiment, considering that the word length is 32 bytes then it may XOR every 32 bytes from these 5120 bytes linearly and the output may be used as seed for the random algorithm generator.

[041] Upon determining the seed, the hypervisor 106 may determine the range which may be equal to a number of sectors in the provisioned thin disk 124 calculated based on equation (1). In an exemplary scenario, for a think disk 124 of same volume as the dangling persistent disk 122 of 10 MB and having a sector size equal to 512 bytes, the number of sectors may be equal to 209715120. Accordingly, the address of the first sector may be 0 and the last sector may be 209871519 on the thin disk 124. Accordingly, the seed and the first and the last sector may be fed into the pseudo-random algorithm based on which the pseudo-random algorithm may start generating the random numbers.

[042] Referring now to **FIG. 2A**, each of the data (X0, X1, X2, X3, X4, X5) may be linearly read from the dangling persistent disk 122 by the background thread. Further, each data read, is written onto the thin disk 124 at an offset sector location of the thin disk 124 determined as the random number generated based on the pseudo-random algorithm. Referring now to **FIG. 2B,** an exemplary diagram of a thin disk 124 storing data in various sector locations is illustrated, in accordance with an embodiment of the present disclosure. In an exemplary embodiment, the thin disk 124 may include sectors T0-T11. As per the exemplary embodiment depicted, linearly read data X0 from the sector S0 of the dangling persistent disk 122 may be written onto an offset sector location of T5 of the thin disk 124 based on a random number generated '5'. Subsequently, the next read data X1 from the sector S1 of the dangling persistent disk 122 may be written onto an offset sector location of T3 of the thin disk 124 based on a random number generated '3'. Accordingly, each of the linearly read data (X0, X1, X2, X3, X4, X5) stored in corresponding the sectors S0-S5 of the dangling persistent disk 122 may be written onto the sectors T5, T3, T0, T4, T2, T1 of the thin disk 124 based on the offset sector locations determined using the generated random numbers 5, 3, 0, 4, 2, 1 generated by the pseudo-random algorithm. In an embodiment, the data linearly read from the dangling persistent disk 122 may be copied at offset sector locations such that there is no data saved in some sectors of the thin disk 124.

[043] Once, all the data from the dangling persistent disk 122 has been linearly read and copied onto the thin disk 124, the hypervisor 106 may delete all the data on the dangling

persistent disk 122. The hypervisor 106 may further unblock the input/output access to the dangling persistent disk 122 upon deletion of the data of the dangling persistent disk 122. The hypervisor 106 may further transfer the written data from the thin disk 124 onto the corresponding persistent disk 122 in a linear manner. In an embodiment, sectors of the thin disk 124 having no data saved may be skipped and data from subsequent sectors of the thin disk 124 may be linearly read to copy the data from the thin disk 124 to the dangling persistent disk 122.

[044] Referring now to **FIG. 2C**, an exemplary diagram 200C of a dangling persistent disk 122 storing data transferred from the thin disk 124 is illustrated, in accordance with an embodiment of the present disclosure. In an exemplary embodiment, the data of the dangling persistent disk 122, copied or written onto the thin disk 124 as depicted in **FIG. 2B,** may be copied back onto the dangling persistent disk 122 in a linear manner. Accordingly, the data in the dangling persistent disk 122 may be deleted and the data copied onto the thin disk 124 at sector locations T0-T5 may be written back to the sector locations S0-S5 of the dangling persistent disk 122 in a linear manner by the background thread.

[045] In an embodiment, in case no data is found to be written onto a sector location T0-T11 of the thin disk 124, the corresponding sector location may be skipped, and data stored in subsequent sectors may be linearly read to be written onto sector location S0-S5 of the persistent disk 122.

[046] Accordingly, the present disclosure allows to effectively secure the data saved in the dangling persistent disk 122 by scrambling the initially saved data in the persistent disk 122 as soon as it is determined to be in a dangling state. The data of the dangling persistent disk 122 may be scrambled based on a pseudo-random algorithm. Accordingly, linearly read scrambled data of the dangling persistent disk 122 would make it an unintelligent data or illegible and hence secure it from any malicious attack.

[047] Referring now to **FIG. 3,** a flowchart of a method 300 for securing data of the persistent disk 122, in accordance with an embodiment of the present disclosure is illustrated. In an embodiment, method 300 may include a plurality of steps that may be performed by the hypervisor 106 to secure data of the persistent disk 122 having a dangling status. It is to be noted that **FIG. 3** is explained in conjunction with **FIG. 1**. Each step of the method 300 may be executed by the hypervisor 106.

[048] At step 302, an I/O request may be determined from an I/O queue corresponding to at least one of the persistent volumes 122. At step 304, the hypervisor 106 may determine a status of the persistent disk 122 based on its associated dangling flag.

[049] In an embodiment, at step 304, the status of the persistent disk 122 may be determined as dangling in case the dangling flag associated to the at least one persistent disk 122 is determined as "TRUE". In an embodiment, the dangling flag may be set based on an association of the persistent disk 122 with corresponding one or more containers 128 in the hypervisor environment 100. In general, a dangling flag corresponding to each of the one or more persistent disks 122 may be initialized as "FALSE". In an embodiment, the dangling flag of the at least one of the persistent disks 122 may be set as "TRUE" when the corresponding one or more containers 128 may become non-operational. Further, in case at step 304, the dangling flag of the at least one of the persistent disks 122 is determined as "FALSE" the hypervisor 106 may continue to provide access of the persistent disk 122 and the next I/O request in the I/O queue corresponding to at least one persistent disk 122 may be read at step 302.

[050] Further, at step 306, upon determination of the dangling flag of the persistent disk 122 as "TRUE", the hypervisor 106 may block input/output access to the dangling persistent disk 122 and may provision a thin disk 124 in the storage 120 having a volume about same or greater than a volume of the dangling persistent disk 122.

[051] Further, at step 308, the hypervisor 106 may activate a background thread to linearly read data corresponding to each of a plurality of sectors of the dangling persistent disk 122.

[052] Further, at step 310, the background thread activated by the hypervisor 106 may be enabled to write the linearly read data onto the thin disk 124 at an offset sector location of the thin disk 124 determined as the random number generated based on a pseudo-random algorithm.

[053] At step 312, the input/output access to the dangling persistent disk 122 may be enabled by the hypervisor.

[054] At step 314, the data stored on the dangling persistent disk 122 may be deleted by the hypervisor 106. At step 316, the data written onto the thin disk 124 may be linearly read and copied back to the dangling persistent disk 122.

[055] Thus, the disclosed method and system try to overcome the various technical problems explained earlier when securing data of at least one persistent disk in a hypervisor environment.

[056] As will be appreciated by those skilled in the art, the techniques described in the various embodiments discussed above are not routine, or conventional, or well-understood in the art. The techniques discussed above provide for securing data of at least one persistent disk in a hypervisor environment.

[057] In light of the above-mentioned advantages and the technical advancements provided by the disclosed method and system, the claimed steps as discussed above are not routine, conventional, or well understood in the art, as the claimed steps enable the following solutions to the existing problems in conventional technologies. Further, the claimed steps bring an improvement in the functioning of the device itself as the claimed steps provide a technical solution to a technical problem.

[058] The specification has described a method and system for securing data of at least one persistent disk in a hypervisor environment. The illustrated steps are set out to explain the exemplary embodiments shown, and it should be anticipated that ongoing technological development will change the manner in which particular functions are performed. These examples are presented herein for purpose of illustration, and not limitation. Further, the boundaries of the functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternative boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed. Alternatives (including equivalents, extensions, variations, deviations, etc., of those described herein) will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein. Such alternatives fall within the scope and spirit of the disclosed embodiments.

[059] 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 (300) of securing data of at least one persistent disk (122) in a hypervisor environment, the method (300) comprising:

determining (302), by a hypervisor (106), a status of the at least one persistent disk (122),

wherein the status of the at least one persistent disk (122) is determined as dangling based on a dangling flag associated to the at least one persistent disk (122), and

wherein the dangling flag is set based on an association of the at least one persistent disk (122) with corresponding one or more containers (128) in the hypervisor environment; and

upon determining the dangling flag of the at least one persistent disk (122) as TRUE:

provisioning (306), by the hypervisor (106), a thin disk of volume about same or greater than a volume of the at least one persistent disk;

reading (308), by the hypervisor (106), data corresponding to each of a plurality of sectors of the at least one persistent disk (122) in a linear manner by a background thread; and

writing (312), by the hypervisor (106), the linearly read data onto the thin disk (124) based on a pseudo-random algorithm.

2. The method (300) as claimed in claim 1, wherein the linearly read data is written onto the thin disk (124) by:

for each read data by the background thread:

generating (310), by the hypervisor (106), a random number based on the pseudo-random algorithm; and

writing (312), by the hypervisor (106), the each read data onto the thin disk at an offset sector location of the thin disk (124) determined as the random number.

3. The method (300) as claimed in claim 2, wherein each of the random number generated is a non-repeated number for a range and a seed of the pseudo-random algorithm,

wherein the range and the seed of the pseudo-random algorithm is determined based on a sector size of the dangling persistent disk (122) and a word length of the data saved in the each of the plurality of sectors of the at least one persistent disk (122).

4. The method (300) as claimed in claim 1, comprising:

blocking, by the hypervisor (106) upon determining of the status of the dangling flag as TRUE, input/output access to the at least one persistent disk (122) until the background thread completes the reading of the data and the writing of the linearly read data onto the thin disk (124).

5. The method (300) as claimed in claim 4, comprising:

deleting, by the hypervisor (106), the data on the at least one persistent disk (122) when the background thread completes the reading of the data and the writing of the linearly read data onto the thin disk (124); and

unblocking, by the hypervisor (106), the input/output access to the at least one persistent disk (122) upon deletion of the data; and

transferring, by the hypervisor (106), the written data from the thin disk (124) onto the at least one persistent disk (122).

6. The method (300) as claimed in claim 1, wherein the dangling flag associated to the at least one persistent disk (122) is set as TRUE when each of the corresponding one or more containers (128) are determined to be non-operational in the hypervisor environment.

7. A system (100) of securing data of at least one persistent disk (122) in a hypervisor environment, 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, cause the processor (114) to enable a hypervisor (106), wherein the hypervisor (106) is configured to:

determine a status of the at least one persistent disk (122),

wherein the status of the at least one persistent disk (122) is determined as dangling based on a dangling flag associated to the at least one persistent disk (122), and

wherein the dangling flag is set based on an association of the at least one persistent disk (122) with corresponding one or more containers (128) in the hypervisor environment; and

upon determining the dangling flag of the at least one persistent disk (122) as TRUE:

provision a thin disk (124) of volume about same or greater than a volume of the at least one persistent disk (122);

read data corresponding to each of a plurality of sectors of the at least one persistent disk (122) in a linear manner by a background thread; and

write the linearly read data onto the thin disk (124) based on a pseudo-random algorithm.

8. The system (100) as claimed in claim 7, wherein the linearly read data is written onto the thin disk (124) based on:

for each read data by the background thread:

generate a random number based on the pseudo-random algorithm; and

write the each read data onto the thin disk (124) at an offset sector location of the thin disk (124) determined as the random number.

9. The system (100) as claimed in claim 8, wherein each of the random number generated is a non-repeated number for a range and a seed of the pseudo-random algorithm,

wherein the range and the seed of the pseudo-random algorithm is determined based on a sector size of the dangling persistent disk (122) and a word length of the data saved in the each of the plurality of sectors of the at least one persistent disk (122).

10. The system (100) as claimed in claim 8, wherein each of the random number generated is a non-repeated number for a range and a seed of the pseudo-random algorithm,

wherein the range and the seed of the pseudo-random algorithm is determined based on a sector size of the dangling persistent disk (122) and a word length of the data saved in the each of the plurality of sectors of the at least one persistent disk (122).

Dated this 22nd day of December 2023

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

ABSTRACT

METHOD AND SYSTEM OF SECURING DATA OF PERSISTENT DISKS

A system (100) and method (300) of securing data of at least one persistent disk (122) in a hypervisor environment is disclosed. A hypervisor (106) determines a status of the at least one persistent disk (122) based on a dangling flag corresponding to the at least one persistent disk (122). Upon determining the dangling flag as TRUE, the hypervisor (106) provisions a thin disk (124) of volume about same or greater than a volume of the at least one persistent disk (122). Further, the hypervisor (106) reads data corresponding to each of a plurality of sectors of the at least one persistent disk (122) in a linear manner by a background thread. Further, the hypervisor (106) writes the linearly read data onto the thin disk (124) based on a pseudo-random algorithm.

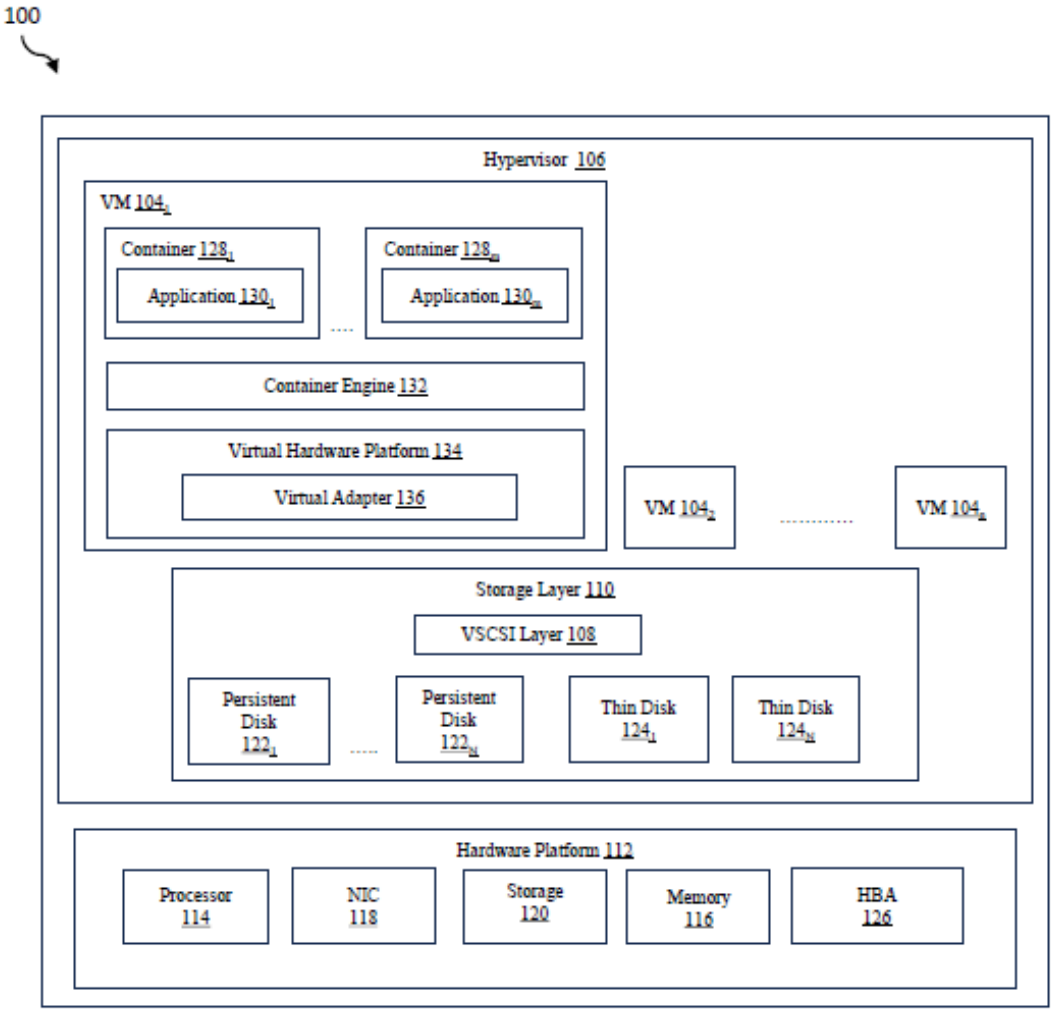

FIG.1

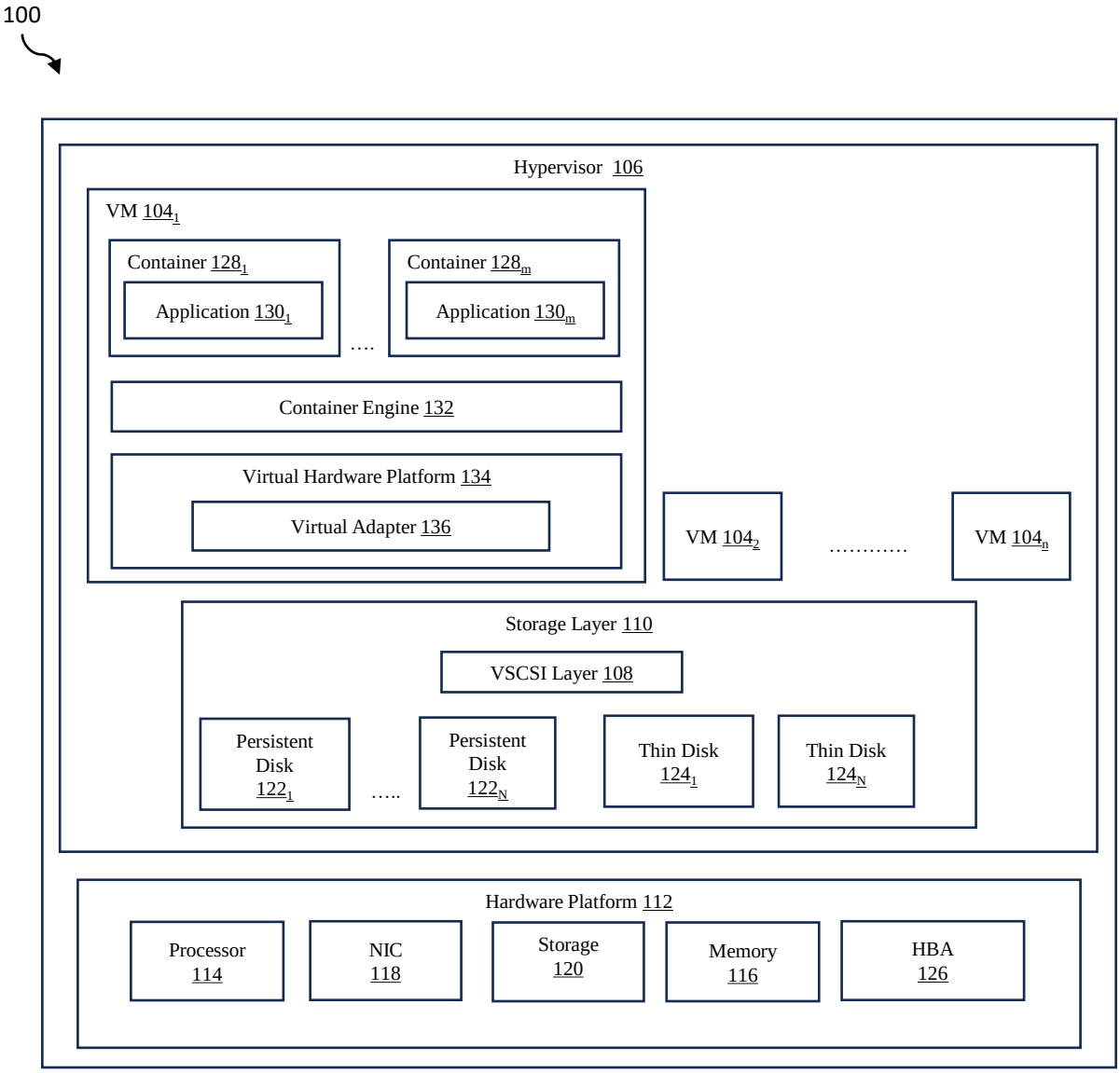

FIG.1

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

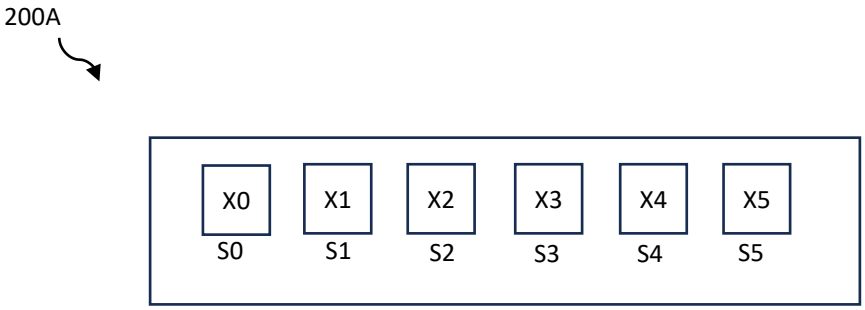

FIG.2A

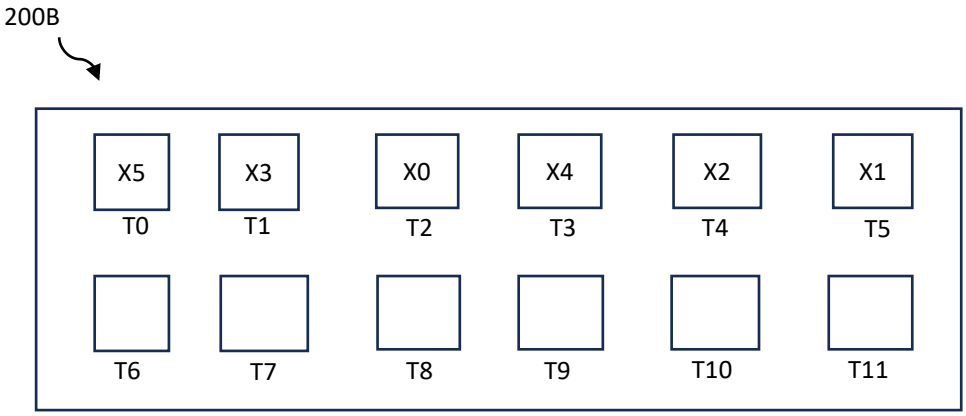

FIG.2B

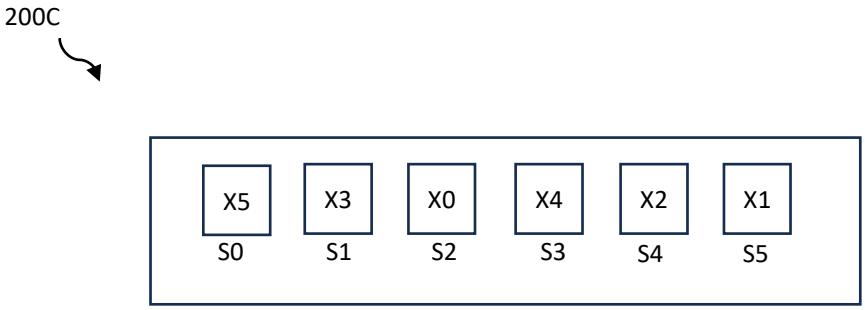

FIG.2C

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

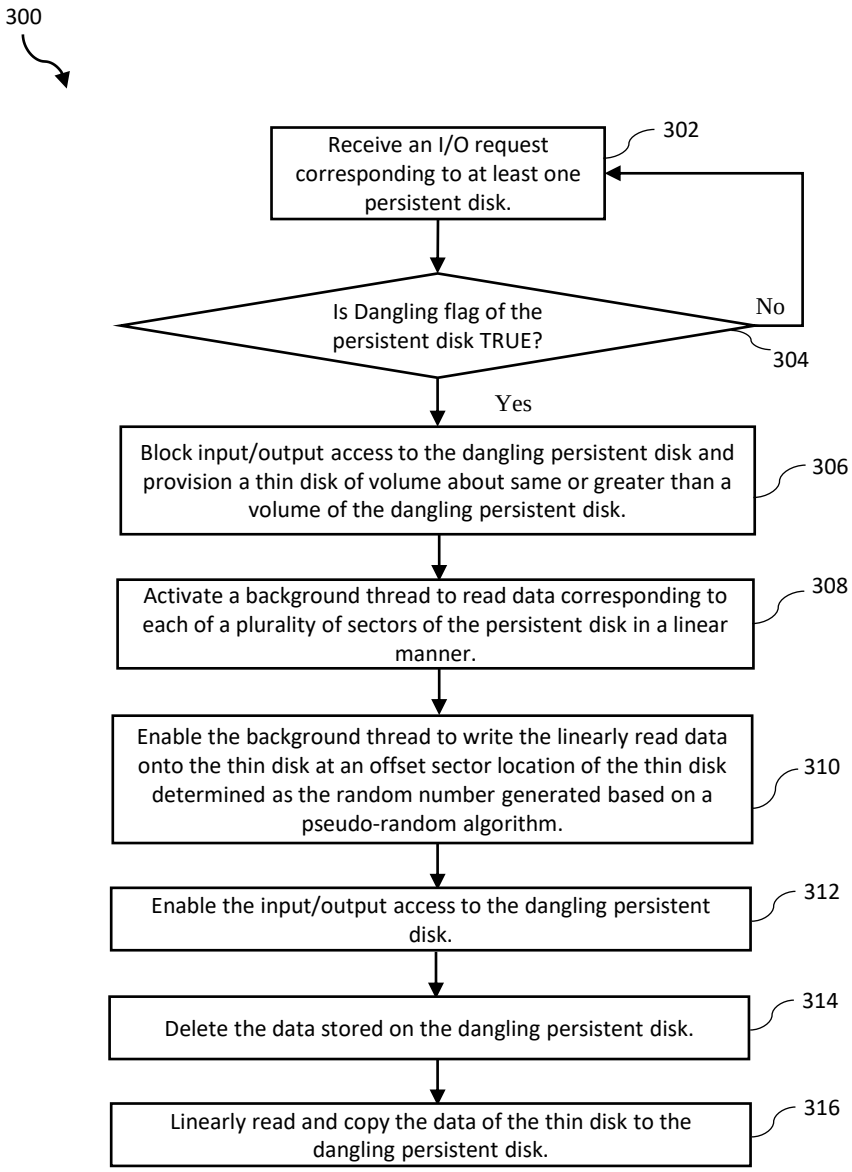

FIG. 3

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