

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
12 March 2009 (12.03.2009)

PCT

(10) International Publication Number  
**WO 2009/033074 A2**

(51) International Patent Classification:

G06F 15/16 (2006.01) A61K 39/42 (2006.01)  
G06F 13/00 (2006.01) C07K 14/00 (2006.01)

(21) International Application Number:

PCT/US2008/075467

(22) International Filing Date:

5 September 2008 (05.09.2008)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

|            |                                |    |
|------------|--------------------------------|----|
| 60/970,187 | 5 September 2007 (05.09.2007)  | US |
| 60/970,181 | 5 September 2007 (05.09.2007)  | US |
| 11/864,583 | 28 September 2007 (28.09.2007) | US |
| 11/864,756 | 28 September 2007 (28.09.2007) | US |

(71) Applicant (for all designated States except US): EMC CORPORATION [US/US]; 176 South Street, Hopkinton, MA 01748 (US).

(72) Inventor; and

(75) Inventor/Applicant (for US only): YUEH, Jedidiah [US/US]; 28 Preston, Irvine, CA 92618 (US).

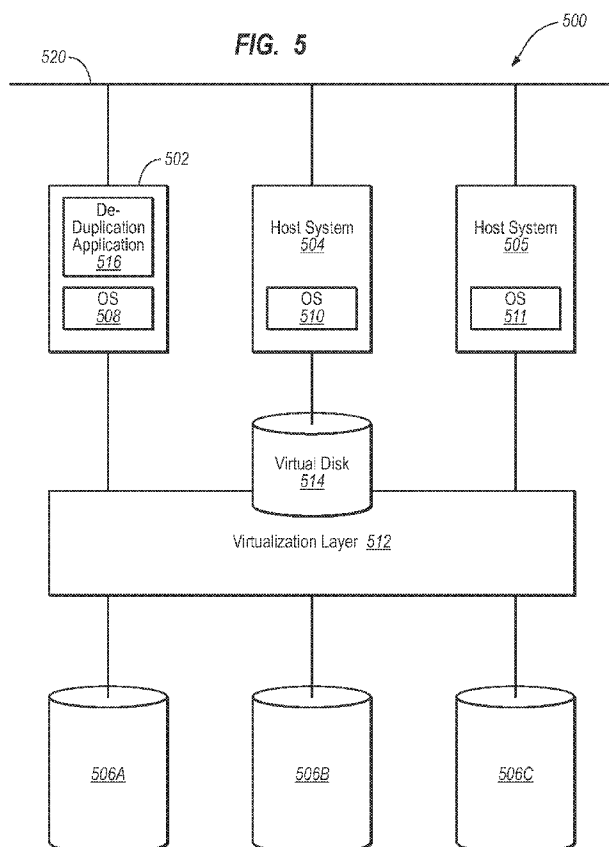
(74) Agents: ISRAELSEN, R., Burns et al.; Workman Nydegger, 1000 Eagle Gate Tower, 60 East South Temple, Salt Lake City, UT 84111 (US).

(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, BR, BW, BY, BZ, CA, CH, 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, IS, JP, KE, KG, KM, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),

[Continued on next page]

(54) Title: DE-DUPLICATION IN VIRTUALIZED SERVER AND VIRTUALIZED STORAGE ENVIRONMENTS



(57) Abstract: A data de-duplication application de-duplicates data on the primary storage read/write pathway of a virtualized server environment and/or in pooled storage capacity of a virtualized storage environment. A virtualized server environment includes multiple server applications operating on a virtualization layer provided on a computer architecture that includes memory for temporarily storing data and storage for persistently storing data. A virtualized storage environment includes multiple storage devices and a virtualization layer that aggregates all or a portion of the storage capacity of each storage device into a single pool of storage capacity. In the virtualized environments, the de-duplication application identifies redundant data in memory, storage, and/or pooled storage capacity and replaces the redundant data with one or more pointers pointing to a single copy of the data. The de-duplication application operates on fixed or variable size blocks of data and de-duplicates data either post-process or in-line.

WO 2009/033074 A2



European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**

— *without international search report and to be republished upon receipt of that report*

## DE-DUPLICATION IN VIRTUALIZED SERVER AND VIRTUALIZED STORAGE ENVIRONMENTS

### BACKGROUND OF THE INVENTION

#### 5 1. The Field of the Invention

The present invention relates to data de-duplication. More particularly, embodiments of the invention relate to software, hardware, systems, and methods for de-duplicating redundant data in virtualized server and virtualized storage environments.

#### 10 2. The Related Technology

Virtualization is an abstraction layer that decouples physical computing resources in a computer environment from systems, applications, and/or end users that interact with those resources to deliver greater IT resource utilization and flexibility. For instance, server virtualization allows multiple virtual machines, with heterogeneous operating systems, to run in isolation, side-by-side on the same physical machine. Each virtual machine has its own set of virtual hardware (e.g., RAM, CPU, NIC, etc.) upon which an OS and applications are loaded. The OS sees a consistent, normalized set of hardware regardless of the actual physical hardware components.

Virtual machines are encapsulated into files, making it possible to rapidly save, copy, and provision a virtual machine. Full systems (fully configured applications, operating systems, BIOS and virtual hardware) can be moved, within seconds, from one physical server to another for zero-downtime maintenance and continuous workload consolidation.

Similarly, storage virtualization is the amalgamation of multiple storage devices into what appears to be a single storage unit. Storage virtualization presents a simple object (such as a volume) upward in a stack to, e.g., a host system, hiding the physical complexity of underlying networks, storage, and other constructs. Storage virtualization can provide many benefits, including centralizing storage management, easier replication, non-disruptive data migration when subsystems fail or are replaced, and implementation of cost-effective tiered storage, to name a few.

Notwithstanding its many advantages, virtualization can result in the unnecessary short- and long-term storage of significant amounts of redundant data in memory (e.g., processor registers, cache memory, RAM, and the like) and/or pooled storage capacity. For instance, to operate fifteen (or more) server applications as fifteen virtual machines in a virtual server environment, fifteen separate operating systems have to be loaded into

memory (e.g., a separate OS is loaded into the memory allocated to each virtual machine), even if two or more of the operating systems are identical and/or include identical data, files, executables, or the like. Similarly, fifteen separate operating systems have to be maintained in persistent/non-volatile storage (e.g., a separate OS is maintained in the storage allocated to each virtual machine) that can include pooled storage capacity.

As another example, consider an email server servicing an electronic message with a large attachment sent to a plurality of intra-system users in a virtualized storage environment. For every user the attachment is sent to, the email server stores a separate copy of the attachment in a portion of the pooled storage capacity allocated to the email server.

In each of the cases just described, redundant instances of data occupy space in memory, storage and/or pooled storage which could be used for other data. Consequently, there currently exists a need in the art for data de-duplication solutions in virtualized server and virtualized storage environments.

The subject matter claimed herein is not limited to embodiments that solve any disadvantages or that operate only in environments such as those described above. Rather, this background is only provided to illustrate one exemplary technology area where some embodiments described herein may be practiced

#### BRIEF DESCRIPTION OF THE DRAWINGS

To further clarify the advantages and features of the present invention, a more particular description of the invention will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. It is appreciated that these drawings depict only typical embodiments of the invention and are therefore not to be considered limiting of its scope. The invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

Figure 1 illustrates a virtualized server environment within which a data de-duplication application can be implemented according to an embodiment of the invention using a virtualization layer to separate the de-duplication application, a server application, and associated operating systems from a computer architecture;

Figure 2 illustrates another virtualized server environment that is similar to that of Figure 1 but arranged as a hosted architecture wherein a data de-duplication application is not virtualized;

Figure 3 depicts yet another virtualized server environment illustrating the use of a virtual infrastructure or layer to allow multiple virtual machines to access a computer resources pool that includes memory and storage;

Figure 4 depicts a method for de-duplicating redundant data in memory and storage of a computer architecture upon which one or more virtualized servers operate  
5 according to one embodiment of the invention;

Figure 5 illustrates a storage virtualization environment in which embodiments of the invention can be implemented;

Figure 6 illustrates a host-based storage virtualization environment according to  
10 an embodiment of the invention;

Figure 7 illustrates a storage device-based storage virtualization environment according to another embodiment of the invention;

Figure 8 illustrates a network-based storage virtualization environment according to yet another embodiment of the invention; and

Figure 9 illustrates a method for de-duplicating data in a virtualized storage  
15 environment.

#### DETAILED DESCRIPTION

Embodiments of the invention relate to a data de-duplication application that de-duplicates redundant data on the primary storage (or simply “storage”) read/write  
20 pathway of a virtualized server environment and/or in the pooled storage capacity of a virtualized storage environment. As used herein, “primary storage” or “storage” refers to hard disk and other persistent storage wherein primary copies of data, such as files and executables, are stored. Prior to being operated on or processed, data from primary storage is read into memory where it can be accessed by a processor such as a central  
25 processing unit (“CPU”). As used herein, “memory” refers to processor registers, cache memory, random access memory (“RAM”) and other temporary memory that can be quickly accessed by a computer’s processor. It is understood that access speeds for memory are usually orders of magnitude greater than access speeds for storage. New and altered data generated by the processor upon operating on and processing data in the  
30 memory is typically first stored in memory and then written from memory to storage for persistent storing.

Server virtualization can be accomplished by providing an abstraction or virtualization layer on a first computer environment including memory for the temporary storage of data and disk or other storage for the persistent storage of data. One or more

server applications are operated on the virtualization layer, each configured to read data from storage into memory and to write data from memory to storage during operation. The virtualization layer presents a representation of resources (such as memory, storage, and the like) within the computer environment to the server applications. Each server application is encapsulated within a virtual machine and provided with an OS to manage  
5 corresponding virtualized hardware and software resources presented to each server application.

A data de-duplication application is operated in the first computer environment to reduce redundant data in memory and/or storage. The de-duplication application  
10 identifies redundant data and replaces it with a reference and/or pointers to a copy of the data that is already present in the memory or storage. For instance, if the OSes used by each of the server applications are identical, a single instance of the OS can be stored in memory when the server applications are booted and the OS for each is loaded. In this case, redundant copies of the OS can be replaced by pointers that point to a single copy of  
15 the OS in memory. Similarly, when redundant data is written to storage, the redundant data can be replaced with pointers that point to a single instance of the data in storage. The de-duplication application can perform de-duplication in-line or post process.

By de-duplicating data on the storage read/write pathways, more data can be read from relatively slow-access storage into relatively fast-access memory. With the  
20 increased amount of data that can be represented in the relatively fast-access memory, access speeds are improved. Additionally, fewer input/output ("I/O") operations to the storage are required to access data since more data can be stored in memory. Further, processing overhead in storage implemented as a Storage Area Network ("SAN") and the like is improved when data written to the storage is de-duplicated inline since less data is  
25 written to the storage.

Alternately, storage virtualization can be accomplished by providing an abstraction or virtualization layer on a second a computer environment including a plurality of storage devices for the storage of data to pool or aggregate some or all of the storage capacity of each storage device. The virtualization layer can be host-based,  
30 storage-based, or network-based. In one embodiment, the virtualization layer presents representations of at least two portions of the underlying storage capacity to at least two host systems. Each of the host systems can generate input/output ("I/O") requests to write data to and/or read data from their corresponding portions of the storage capacity.

Each I/O request includes a virtual memory address that the virtualization layer maps to a physical memory address.

A data de-duplication application is operated in the second computer environment to reduce redundant data in the aggregated storage capacity. The de-duplication application identifies redundant data in the storage capacity and replaces it with a reference/pointer to a copy of the data that is stored elsewhere in the storage capacity. For instance, the host systems may implement identical operating systems (“OSes”) that include many identical files, executables, or other blocks of data. In this case, the data de-duplication application identifies the identical blocks of data, eliminates redundant copies from the storage capacity, and stores pointers at the former locations of the redundant copies pointing to a single remaining instance of the data.

To practice the invention, the computer and network devices may be any devices useful for providing the described functions, including well-known data processing and storage and communication devices and systems such as computer devices typically used as hosts in user systems with processing, memory, and input/output components, and server devices configured to maintain and then transmit digital data over a communications network. Data typically is communicated in digital format following standard communication and transfer protocols. The data storage resources are generally described as disk, optical, and tape devices that implement RAID and other storage techniques and that may use SCSI and other I/O, data transfer, and storage protocols, but the invention is not intended to be limited to the example embodiments or to specific hardware and storage mechanisms as it is useful for nearly any computer environment.

## **I. VIRTUALIZED SERVER ENVIRONMENTS**

Figure 1 illustrates one embodiment of a virtualized server system 100 in which embodiments of the invention may be implemented. As shown, the virtualized server system 100 is built on a computer platform 110 (e.g., the x86 computer architecture or the like) that includes a number of resources such as CPUs, memory, network devices, disk, and tape mechanisms. A virtualization layer 120 is provided to manage access to or “virtualize” the architecture 110 into a uniform pool and separate the hardware of this pool from one or more virtual machines (i.e., the system 100 is shown with 2 such virtual or guest machines running on the virtualization layer 120 - but more such machines may be run). The virtualization layer 120 presents a virtualized representation 136, 137 to an OS 132, 133 of each virtual or guest machine. The operating systems 132, 133 may be

the same or vary and may include for example, Windows, Linux, Novell, Solaris, and other operating systems such as FreeBSD and the like.

A server application 131 is provided and associated with the OS 133 and uses the virtual system 137 to read data into memory from storage and/or to write data to storage from memory. The virtual system 137 shields the server application 131 from the actual resources 110, presenting, for instance, a logical representation of all or a portion of the memory and all or a portion of the disk storage to the server application. While the virtualized server system 100 is illustrated with a single server application 131 running on the virtualization layer 120, in another embodiment of the invention the virtualized server system 100 may have a plurality of server applications 131 running on the virtualization layer 120, each provided with an OS 133 and standardized hardware 137.

Additionally, a data de-duplication application 130 is provided and associated with the OS 132 and uses the virtual system 136 to process data read into memory or written to disk storage to eliminate redundant data within the memory or disk storage, respectively, of the resource pool 110. The de-duplication application 131 can be virtually any de-duplication application configured to identify redundant data and replace the redundant data with one or more pointers or references pointing to a single instance of the data. For instance, in one embodiment of the invention the de-duplication application 131 de-duplicates data in-line. In other words, the de-duplication application identifies and replaces redundant data with pointers as data is ingested, read, or written into memory or storage. According to another embodiment of the invention, the de-duplication application 131 de-duplicates data post-process. In this example, the de-duplication application identifies and replaces redundant data with pointers after the data has already been ingested, read, or written into memory or storage. Additionally, the de-duplication application can de-duplicate data using a fixed block size or a variable block size. While many data de-duplication algorithms are known in the art and can be implemented by the de-duplication application 130, in one embodiment the de-duplication application implements the sticky byte algorithm disclosed in commonly assigned U.S. Patent No. 6,810,398, which is incorporated herein by reference.

Generally, the virtualization layer 120 is selected to act as an abstraction layer that decouples the physical hardware from the OS associated with applications 130, 131 to deliver greater IS resource utilization and flexibility. Virtualization layer 120 allows multiple virtual machines or guest machines (such as may be used to run applications 130, 131), with homogeneous or heterogeneous operating systems, to run in isolation but

side-by-side on the same or a different physical machine. Each virtual machine has its own set of virtual hardware (e.g., RAM, CPU, NIC, etc. within the pool 110) upon which an OS and applications including the applications 130, 131 are loaded. The operating systems see a consistent, normalized set of hardware regardless of the actual physical hardware components.

Advantageously, virtual machines (such as the two virtual machines of Figure 1) can be encapsulated into files, making it possible to rapidly save, copy and provision a virtual machine. Full systems (fully configured applications, operating systems, BIOS and virtual hardware) can be moved, within seconds, from one physical server to another for zero-downtime maintenance and continuous workload consolidation. Additionally, the virtualization layer 120 provides isolation. Virtual machines are completely isolated from the physical host machine and other virtual machines. If a virtual machine crashes, all others are unaffected. Data does not leak across virtual machines and applications can only communicate over configured network connections.

Figure 2 illustrates an alternative arrangement for a virtualized server system 200 according to embodiments of the invention. As with virtualized server system 100, the virtualized server system 200 includes a set of computer hardware or a particular architecture 210. In this example, a de-duplication application 222 is installed and run in the architecture 210 as a conventional application, relying on a host OS 220 for device support and physical resource management of the architecture 210. The system 200 additionally includes a virtual machine that is installed and run as an application, e.g., similar to application 222. The virtual machine includes the virtualization layer 230 that shields a server application 238 within the virtual machine from the computer resources 210. As with the application 222, the virtualization layer 230 relies on the host OS 220 for device support and physical resource management of the computer hardware 210. Additionally, the virtualization layer 230 presents a guest OS 234 to the server application 238.

Figure 3 illustrates yet another embodiment of a virtualized system 300 according to the invention. The virtualized system 300 includes a computer resources pool or computer architecture/platform 310 that includes the hardware and associated software available in the virtualized system 300. The pool 310 includes primary storage 311, 318 that may include nearly any type of storage device for digital data such as disk devices. The pool 310 also includes networks/network devices 312, 314 and a number of physical servers or other computing devices 313, 315, 316 (which may also be used as data storage

in some cases with storage 311, 318 being used additionally or alternately for archive or backup storage).

Upon the platform 310, a virtualization infrastructure 320 is provided for connecting the resources in pool 310 to users (or to a business/enterprise) 335, 345, 355. The virtual infrastructure 320 provides a dynamic mapping of the resources in pool 310 to one or more virtual machines 330, 340, 350. Each of the virtual machines 330, 340, 350 runs an OS 332, 342, 352 and an application 334, 344, 354. In the present embodiment of the invention, each of the applications 334 and 344 is a server application whose operation includes reading data into virtual memory from virtual storage and/or writing data into virtual storage from virtual memory.

The application 354 is a data de-duplication application that replaces redundant data in virtual memory and/or in virtual storage with a pointer or reference pointing to a single instance of the data. As previously discussed, this de-duplication process can occur in-line or post-process using a fixed or variable block size. Further, it can operate within a virtual machine 330 on the virtual infrastructure 320, as illustrated, or it can be installed and operated as a conventional application.

De-duplicating data in memory and/or storage increases the amount of available space in memory and/or storage since only a single instance of data is stored in memory and/or storage, respectively. As a result of this effective increase in size of memory, more data can be read into memory, thereby improving access speeds. Additionally, the amount of storage I/O is reduced since a greater amount of data is stored in memory. When data is de-duplicated in-line prior to being written to storage, less data is written to storage, which reduces processing overhead in storage such as a SAN, and the like. Additionally, a greater amount of data can be stored in the storage when the data is de-duplicated.

With reference now to Figure 4, a method 400 is illustrated for de-duplicating data in memory and/or storage of a computer architecture such as the virtualized environments of Figures 1-3. A typical computer architecture in which the method 400 may be practiced includes one or more memory devices and one or more storage devices, wherein data can be read from storage into memory and/or can be written from memory to storage. The process 400 begins by providing 402 a virtualization layer on the computer architecture. One or more server applications can then be run or operated 404 on the virtualization layer, in the process of which the server application reads data into memory and writes data to storage. The virtualization layer presents a representation of at least a

portion of the memory and at least a portion of the storage to the server application wherein data used by the server application can be stored.

A data de-duplication application can also be operated or run 406 on the virtualization layer or on the operating system of a physical host machine to de-duplicate  
5 redundant data in the memory and/or storage. When operated or run on the virtualization layer, the virtualization layer presents a representation of a set of components of the computer architecture to the de-duplication application such that the de-duplication application can de-duplicate redundant data.

Upon running the de-duplication application, the application de-duplicates 408  
10 data in memory, storage, or both. Note that when de-duplicating data, the de-duplication application is configured to view the memory and storage as separate systems. Thus, if a single instance of data is stored in both memory and in storage, the de-duplication application does not eliminate one of the instances. Instead, the de-duplication application is configured to eliminate redundant data in memory relative only to the data  
15 stored in memory and/or to eliminate redundant data in storage relative only to the data in storage.

In one embodiment, de-duplicating 408 data includes identifying redundant data, deleting it, and replacing it with a pointer that points to a single instance of the deleted data. Although various methods and algorithms can be implemented according to the  
20 invention, in one embodiment identifying redundant data comprises breaking data into one or more pieces of a variable or fixed size, performing a hash on each piece to obtain a hash value, and comparing the resulting hash value to hash values for data already present in the system. If the de-duplication application implements a deterministic hash function that produces probabilistically unique hash values (e.g., SHA-1, MD5, and the like), then  
25 identical hash values indicate that the underlying data is identical. Additionally, hash values can be used as the references or pointers that point to single instances of data.

Returning to Figure 4, the server application can access 410 de-duplicated data by reading the pointers, which refer the server application to the single instance of the data.

One skilled in the art will appreciate that data de-duplication according to the  
30 invention can be implemented across server application virtual machines operating on a single physical server, or across server application virtual machines operating across a plurality of physical servers.

As discussed above, the virtualization layer or infrastructure (such as those shown in Figures 1-3) may take a number of forms to practice the invention. In one

embodiment, however, the server applications and optionally the de-duplication application may be run on VMware virtualization products such as VMware ESX or GSX (or similar products). Virtualization products such as VMware ESX are relatively well known and understood by those skilled in the art and a full discussion is not believed  
5 required here to describe the invention. In other embodiments, the virtualization layer is provided by the virtualization product Xen 3.0 (or other versions) from XenSource, and in other embodiments, virtualization products from Microsoft Corporation and/or Intel Corporation are used to provide the virtualization layer described herein.

## II. VIRTUALIZED STORAGE ENVIRONMENTS

10 Figure 5 illustrates an example of a virtualized storage environment 500 in which embodiments of the invention may be implemented. As shown, the virtualized storage environment 500 includes a plurality of host systems 502, 504, 505 and a plurality of storage devices 506A, 506B, and 506C (collectively referred to as "storage devices 506"). The host systems 502, 504, 505 are computer systems that run one or more server  
15 applications, such as a file server, email server, and the like, or any combination thereof. Operating systems 508, 510 and 511 are installed and run on the host systems 502, 504, 505 to manage the sharing of the resources of the host systems amongst the one or more server applications run on the host systems. In the example of Figure 5, host systems 502, 504, 505 can exchange data over network 520, typically a local area network  
20 ("LAN"), wide area network ("WAN"), and the like or any combination thereof. Although not illustrated, network 520 can additionally provide a communication path for various client computer systems to communicate with servers 502, 504, 505.

The storage devices 506 may comprise, for instance, tape drives, disk drives, optical drives, storage arrays, and the like or any combination thereof. Additionally, the  
25 storage devices 506 can be directly attached to the host systems 502, 504, 505 in a direct attached storage ("DAS") configuration, they can be attached to the host systems 502, 504, 505 through a network in a network attached storage ("NAS") configuration, or they can be attached to the host systems 502, 504, 505 via a storage area network ("SAN").

An abstraction or virtualization layer 512 is provided in the virtualized storage  
30 environment 500 to manage access to or "virtualize" some or all of the storage capacity from each of the storage devices 506 into a uniform pool and separate the hardware of this pool from the host systems 502, 504, 505. For instance, the virtualization layer 512 presents a virtualized representation 514 of at least a portion of the storage devices 506 to the host system 504. One skilled in the art will appreciate that the virtualized

representation 514 can be mapped by the virtualization layer to physical storage on all or a portion of a single storage device (such as the device 506A) or to physical storage on all or portions of multiple storage devices (such as on at least two of devices 506A, 506B and 506C).

5         In this example, the host system 504 sends various I/O requests to perform various operations on storage system 506. Such I/O requests typically include read and write requests. Both read requests and write requests include a virtual memory address, while write requests additionally include data to be written to the specified address. The virtualization layer 512 receives I/O requests from the host system 502 and maps virtual  
10         storage addresses to physical storage addresses. For instance, for a read request the virtualization layer sends the read request to the physical storage address and retrieves the data stored at the location, after which the data can be returned to the host system 504 as if it had come from the specified virtual storage address. For a write request, the virtualization layer 512 sends the data to the specified physical storage address for  
15         storage.

       A data de-duplication application 516 is provided in the virtualized storage environment 500 communicatively coupled to the storage devices 506 via the virtualization layer 512. The de-duplication application 516 is illustrated as operating on server or host system 502 but de-duplication agents can also be installed in storage  
20         devices (e.g., in a storage array) 506 and/or servers 502, 504. De-duplication application 516 processes data stored in storage devices 506 to identify and reduce redundant data. The de-duplication application 516 can be virtually any de-duplication application configured to identify redundant data and replace the redundant data with one or more pointers or references pointing to a single instance of the data, as already described above  
25         with respect to the de-duplication application 131 of Figure 1. More particularly, the de-duplication application 516 can de-duplicate data in-line, post-process, using a fixed block size, using a variable block size, and/or can implement the sticky-byte algorithm disclosed in U.S. Patent No. 6,810,398.

       The de-duplication application 516 can de-duplicate data in all or only a portion of  
30         the aggregate storage capacity of the storage devices 506 by configuring the virtualization layer 512 to present a representation of all or only a portion of the pooled storage capacity to the de-duplication application 516. For instance, a portion of the aggregated storage can be allocated as a backup storage target. In this case, it may be desirable to prevent the de-duplication server from considering the storage allocated for backups when

performing de-duplication. Alternately, it may be desirable to present a representation of all of the shared storage capacity to the de-duplication application 516 in order to globally de-duplicate data stored in the aggregate storage capacity.

Embodiments of the invention enable efficient utilization of the pooled storage capacity of storage devices 506 due to the de-duplication of redundant data. Further, operating in a virtualized storage environment enables one or more of the following benefits. For instance, storage capacity from multiple vendors can be combined for centralized management. Storage utilization can be increased by providing more flexible access to storage assets. Administrative costs can be decreased by enabling management of pooled storage from a single interface. Host applications (e.g., server applications running on the host systems 502, 504) are insulated from changes to the physical storage infrastructure. Data can be migrated among storage systems without interrupting applications. And finally, common network-based copy services can be applied across storage systems 506 from multiple vendors.

Figure 5 depicts a virtualized storage environment 500, without illustrating the particular methodology of virtualization. In Figures 6 through 8, three different virtualization implementation methodologies are illustrated: host-based, storage-based, and network-based. The principles of the invention can be implemented in conjunction with either of these three methodologies, as well as with other methodologies and variations thereof. It is understood that the virtualized storage environments of Figures 6-8 are examples only, and should not be construed to limit the invention.

Figure 6 illustrates one embodiment of virtualized storage environment 600 in which storage virtualization is implemented using a host-based methodology. According to this embodiment, the virtualized storage environment 600 includes a plurality of host systems 602, 604, and 605 communicatively coupled via network 620. The network 620 may also be configured to provide a communication path for various client computer systems to communicate with host systems 602 and 604.

The virtualized storage environment 600 additionally includes a SAN 630, switches 608 and 610, and storage devices 606A and 606B, implemented here as a pair of storage arrays. Each of the storage arrays 606 may include, for instance, a controller and one or more disk drives. The SAN 630 is typically a high-speed network that allows the establishment of direct connections between the storage devices 606 and the servers 602, 604. The SAN 630 can be implemented using a variety of different technologies,

including Fibre Channel arbitrated loop, Fibre Channel switched fabric, IP networks (e.g., iSCSI), Infiniband, and the like.

According to this embodiment of the invention, each host system 602, 604, and 605 relies on a virtualization agent 612, 614, and 615 to provide the virtualization layer  
5 between the storage devices 606 and the host systems 602, 604, 605. The virtualization agents 612, 614, 615 monitor I/O traffic and storage 606 to perform the mappings between virtual storage addresses presented to the host systems 602, 604, 605 and physical storage addresses within the storage devices. Mapping data (also referred to herein as “virtualization metadata” or simply “metadata”) can be exchanged between  
10 virtualization agents 612, 614, 615 to enable global storage management. The host-based virtualized storage environment 600 enables low-cost initial deployment and incremental scaling. Further, host-based virtualization allows administrators to access advanced storage management functions such as mirroring, RAID sets, redundant pathing, and hot backups (by using mirror splits or snap-shots).

15 According to the present embodiment of the invention, a de-duplication application 616 is operated on the host system 602 to provide data de-duplication within the portion of the pooled storage capacity presented to the de-duplication application by the virtualization agent 612.

With reference now to Figure 7, an example of a virtualized storage environment  
20 700 is illustrated in which storage virtualization is implemented using a storage-based methodology. The virtualized storage environment 700 includes host systems 702, 704, 705 communicatively coupled via network 720. SAN 730 and switch 732 enable the establishment of direct connections between a storage platform 740 and the servers 702, 704, 705. Storage platform 740 is additionally coupled to external storage arrays 742 and  
25 744.

Each of the storage platform 740 and storage devices 742, 744 has a storage controller 746, 748, and 749 configured to manage an array of physical disk drives 752, 754, and 756 within the platform 740 and storage devices 742, 744, respectively. In this embodiment of the invention, a virtualization layer 750 is implemented in the storage  
30 platform 740. The virtualization layer 750 makes the disk drives 754 and 756 in external storage arrays 742, 744 appear to the host systems 702, 704 as if they were internal to storage platform 740, similar to internal disk drives 752. While the virtualization layer is illustrated in Figure 7 within the storage array 740, storage-based virtualization can

alternately be implemented with a virtualization layer on a standalone controller, rather than on a storage platform as shown.

In the configuration of Figure 7, the controller 746 and virtualization layer 750 pool all the attached storage and present it as a single pool to the host systems 702, 704, 705. Advantageously, all processing of I/O requests is handled off-network and the virtualization metadata can be maintained in internal storage. Further, storage management is centralized at the storage platform 740 and there is no need to exchange metadata between two different locations, as can occur in the virtualized storage environment 600 of Figure 6 in order to enable global storage management. Storage virtualization according to the embodiment of Figure 7 can allow for movement between different RAID groups without data loss, as well as automatic migration of data from one RAID group to another based upon the frequency of data access. In addition, the embodiment of Figure 7 typically permits the creation of multiple data mirrors, which provide additional availability when one of the mirrors is split for hot backups. Storage-based virtualization can provide the flexibility to modify LUN size, the ability to have multiple hosts see the same LUNs, and remote replication.

A data de-duplication application 710 is operated on the host system 702 to provide data de-duplication within the portion of the pooled storage capacity presented to the de-duplication application 710 by the storage platform 740.

With reference now to Figure 8, an example of a virtualized storage environment 800 is illustrated in which storage virtualization is implemented using a network-based methodology. The virtualized storage environment 800 includes host systems 802, 804, 805 communicatively coupled via network 820. SAN 830 and switch 832 enable the establishment of direct connections between storage arrays 840, 842 and the host systems 802, 804, 805. Although not illustrated, each of the storage arrays 840, 842 may additionally include storage controllers and arrays of physical drives, similar to the storage controllers 748, 749 and arrays of disk drives 754, 756 of Figure 7.

In the embodiment of Figure 8, a virtualization layer 850 is implemented in the switch 832. Alternately or additionally, the virtualization layer can be implemented in a dedicated appliance. The switch and/or appliance monitors the SAN 830 for all available storage and builds metadata about stored data and its location. Accordingly, the switch 832 pools the attached storage 840, 842 to present a normalized set of storage hardware to the host systems 802, 804, 805.

There are two kinds of network-based virtualization products: in-band (illustrated in Figure 8) and out-of-band (not illustrated). An in-band virtualization appliance or switch is physically located between the host systems 802, 804 and the storage 840, 842. The appliance or switch takes the disk requests from the host systems and fulfills the host request from the storage attached to the other side of the appliance or switch. This functionality is essentially transparent to the host because the switch or appliance presents itself as disk. The physical location of the appliance or switch is the primary difference between out-of-band and in-band appliances or switches. Out-of-band appliances logically present themselves as if they are located in the data path between the host and storage, but they actually reside outside of the data path. This is accomplished with the installation of a “thin” virtualization driver on the host systems in the I/O data path. The out-of-band appliance provides the virtualization driver with the storage mappings. The virtualization driver presents virtual storage volumes to the applications and file systems on the host and sends the blocks of data directly to correct physical locations on the disks. In contrast, the in-band appliance or switch requires no host-side changes. It acts as a surrogate for a virtual storage volume and performs mapping and I/O direction in a device or computer system located outside of the host.

A data de-duplication application 810 is operated on the host system 802 to provide data de-duplication within the portion of the pooled storage capacity presented to the de-duplication application 810 by the switch 832.

De-duplicating data in pooled storage increases the amount of available storage capacity in virtualized storage devices, which increases the amount of data that can be stored in the virtualized storage devices. Further, when data is de-duplicated in-line by a de-duplication application prior to being written to storage, less data is written to storage, thereby reducing processing overhead for a storage controller in a storage array, as well as reducing network traffic.

With reference now to Figure 9, a method 900 is illustrated for de-duplicating data in a virtualized storage environment, such as the virtualized storage environments 500, 600, 700, 800 of Figures 5-8. A typical computer architecture or environment in which the method can be implemented includes one or more host systems or server machines, and one or more storage devices, wherein the host systems send various I/O requests to perform various operations on the shared storage. The process 900 begins by providing 902 a virtualization or abstraction layer on the one or more storage devices to pool/aggregate some or all of the storage capacity of each storage device into a single

storage pool. The virtualization or abstraction layer presents a representation of at least a portion of the storage pool to one or more host systems for storing data therein.

The one or more host systems are operated 904 within the computer environment and send various I/O requests to the storage pool, such as read and/or write requests.

5 Each read and write request includes a virtual storage address, corresponding to a location in the storage pool allocated to the originating host system. The virtualization or abstraction layer intercepts 906 read and/or write requests and maps virtual storage addresses to physical storage addresses.

The method continues by operating 908 a data de-duplication application to

10 reduce redundant data in the pooled storage capacity. The de-duplication application can be installed and operated on one of the one or more host systems. Operating the de-duplication application includes, in one embodiment, identifying redundant data, deleting it, and replacing it with a pointer that points to a single instance of the data.

Identifying redundant data may comprise breaking data into one or more pieces of

15 a variable or fixed size, performing a hash function on each piece to obtain a hash value, and comparing the resulting hash value to hash values for data already stored in the pooled storage. As explained above, if the de-duplication application implements a deterministic hash function that produces probabilistically unique hash values, then identical hash values indicate that the underlying data is identical. Thus, the described

20 algorithm can be used by the de-duplication application to determine that first data is identical to second data. Upon making this determination, the de-duplication application can replace the redundant data with one or more pointers pointing to a single instance of the data stored in the pooled storage. In one embodiment, the pointers implemented by the data de-duplication application may comprise corresponding hash values obtained

25 during the step of identifying redundant data.

Returning to Figure 9, the one or more host systems can access 910 de-duplicated data by reading pointers, which refer the one or more host systems to the single instance of the data.

The embodiments described herein may include the use of a special purpose or

30 general-purpose computer including various computer hardware or software modules, as discussed in greater detail below.

Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that can be

accessed by a general purpose or special purpose computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired and wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of computer-readable media.

Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

As used herein, the term "module" or "component" can refer to software objects or routines that execute on the computing system. The different components, modules, engines, and services described herein may be implemented as objects or processes that execute on the computing system (e.g., as separate threads). While the system and methods described herein are preferably implemented in software, implementations in hardware or a combination of software and hardware are also possible and contemplated. In this description, a "computing entity" may be any computing system as previously defined herein, or any module or combination of modules running on a computing system.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

## CLAIMS

What is claimed is:

1. A method for de-duplicating redundant data in memory, storage, or both, comprising:
  - 5 providing a virtualization layer on a computer architecture comprising memory for temporarily storing data, and storage for persistently storing data;  
running a server application on the virtualization layer, the server application reading data from a portion of the storage into a portion of the memory and writing data to the portion of the storage from the portion of the memory, the  
10 virtualization layer presenting a representation of at least the portion of the memory and at least the portion of the storage to the server application; and  
running a data de-duplication application on the computer architecture to de-duplicate data in one or more of memory and storage.
2. The method of claim 1, wherein the memory comprises one or more of  
15 processor registers, cache memory, and random access memory and the storage comprises disk storage.
3. The method of claim 1, wherein de-duplicating data in memory comprises:  
identifying first data in memory that is identical to second data in memory;  
deleting the second data from memory; and  
20 replacing the second data with a pointer pointing to the identical first data.
4. The method of claim 3, wherein identifying first data in memory that is identical to second data in memory includes performing a hash function on each of the first and second data, comparing the resulting hash values for the first and second data to each other, and determining that the hash values are identical.
- 25 5. The method of claim 4, wherein the pointer pointing to the identical first data comprises the hash value of the second data.
6. The method of claim 1, wherein de-duplicating data in storage comprises:  
identifying first data in storage that is identical to second data in storage;  
deleting the second data from storage; and  
30 replacing the second data with a pointer pointing to the identical first data.
7. The method of claim 6, wherein identifying first data in storage that is identical to second data in storage includes performing a hash function on each of the first and second data, comparing the resulting hash values for the first and second data to each other, and determining that the hash values are identical.

8. The method of claim 7, wherein the pointer pointing to the identical first data comprises the hash value of the second data.

9. The method of claim 1, wherein the server application comprises a plurality of server applications, the virtualization layer presenting a representation of a separate portion of the memory and a separate portion of the storage to each of the server applications.

10. The method of claim 1, wherein running the data de-duplication application on the computer architecture comprises running the data de-duplication application on the virtualization layer.

11. The method of claim 1, wherein:  
the data de-duplication application de-duplicates data in memory either as the data is read into memory, or after the data has been read into memory; and  
the data de-duplication application de-duplicates data in storage either as the data is written to storage, or after the data has been written to storage.

12. The method of claim 1, wherein de-duplicating data in one or more of memory and storage comprises breaking the data into blocks, the size of the blocks being either fixed or variable.

13. A method for de-duplicating data in a computer environment including one or more virtualized server applications, the method comprising:

providing a virtualization layer on a computer architecture that includes memory for temporarily storing data, and storage for persistently storing data;

operating at least one server application on the virtualization layer, the at least one server application configured to read data from storage into memory and to write data from memory to storage during operation;

presenting a virtualized representation of the computer architecture to the at least one server application;

operating a data de-duplication application on the computer architecture to perform one or more of:

an act of identifying data read into memory that is identical to data already stored in memory; and

an act of identifying data written to storage that is identical to data already stored in storage; and

replacing one or more of:

identical data read into memory with one or more pointers pointing to the corresponding data already stored in memory; and

identical data written to storage with one or more pointers pointing to the corresponding data already stored in storage.

5           14.    The method of claim 13, wherein providing a virtualization layer on a computer architecture further comprises pooling resources of the computer architecture.

          15.    The method of claim 13, wherein the de-duplication application is encapsulated within a virtual machine running on the virtualization layer and wherein the virtualization layer presents a virtualized representation of the computer architecture to an  
10   operating system of the virtual machine.

          16.    The method of claim 13, wherein the computer architecture comprises one or more networks, one or more network devices, or one or more networks and one or more network devices and wherein virtualization layer comprises a virtual infrastructure that connects resources in the computer architecture to users and wherein the virtual  
15   infrastructure provides a dynamic mapping of the resources to a plurality of virtual machines.

          17.    The method of claim 16 wherein the at least one server application comprises a plurality of server applications, each server application being encapsulated within a different one of the plurality of virtual machines and wherein each of the  
20   plurality of virtual machines runs an operating system and wherein at least two of the operating systems are identical.

          18.    The method of claim 17, wherein operating the plurality of server applications includes reading the at least two identical operating systems from storage into memory, and wherein replacing identical data read into memory with one or more  
25   pointers pointing to the corresponding data already stored in memory comprises replacing at least one copy of the at least two identical operating systems read into memory with pointers to a single remaining copy of the at least two identical operating systems.

          19.    The method of claim 13, wherein identical data is replaced with one or more pointers either post-process or in-line.

30           20.    The method of claim 13, further comprising accessing identical data no longer stored in memory, or storage, by using the one or more pointers to access the corresponding data already stored in memory, or storage, respectively.

          21.    A method for de-duplicating redundant data in a virtualized storage environment, the method comprising:

pooling storage capacity from a plurality of storage devices by applying an abstraction layer to the plurality of storage devices, the abstraction layer presenting a representation of a corresponding portion of the pooled storage capacity to each of one or more host systems;

5           operating the one or more host systems in a computer architecture that includes the plurality of storage devices, each host system configured to write data to and read data from a corresponding portion of the pooled storage capacity; and

          operating a data de-duplication application in the computer architecture to de-duplicate data in the pooled storage capacity.

10           22.     The method of claim 21, wherein the abstraction layer is implemented: in at least one of the one or more host systems, in at least one of the plurality of storage devices, or in a switch or other appliance of a storage area network to which the plurality of storage devices are coupled.

          23.     The method of claim 21, wherein de-duplicating data in the pooled storage  
15    capacity comprises:

          identifying first data in the pooled storage capacity that is identical to second data in the pooled storage capacity;

          deleting the second data from the pooled storage capacity; and

          replacing the second data with a pointer pointing to the identical first data.

20           24.     The method of claim 23, wherein identifying first data in the pooled storage capacity that is identical to second data in the pooled storage capacity includes performing a hash function on each of the first and second data, comparing the resulting hash values for the first and second data to each other, and determining that the hash values are identical.

25           25.     The method of claim 24, wherein the pointer pointing to the identical first data comprises the hash value of the first or second data.

          26.     The method of claim 21, wherein the data de-duplication application de-duplicates data written from the one or more host systems to the pooled storage capacity either post-process or in-line.

30           27.     The method of claim 21, wherein the virtualization layer intercepts read and write requests from the one or more host systems and maps virtual storage addresses included in the read and write requests to physical storage addresses in the pooled storage capacity.

28. A method for providing storage virtualization and data de-duplication in a computer environment including a plurality of storage devices, the method comprising:

5       applying a virtualization layer to a plurality of storage devices to aggregate their storage capacity such that during operation, when a write request comprising a virtual memory address and write data is received, the virtualization layer maps the virtual memory address to a physical memory address within the aggregated storage capacity, wherein the aggregated storage capacity includes previously stored data including a first data block;

10       determining that a second block of data included in the write data is identical to the first block of data; and

      storing a pointer in the aggregated storage capacity at the physical memory address instead of storing the second block of data, the pointer pointing to the first block of data.

29. The method of claim 28, further comprising, in response to receiving a mapped read request from the virtualization layer for the second data including the second block of data stored at the physical memory address, using the pointer to access and return the first block of data that is identical to the second block of data.

30. The method of claim 28, further comprising operating a host system in a computer environment that includes the plurality of storage devices, the host system sending the write request to the virtualization layer.

31. The method of claim 28, wherein determining that the second block of data is identical to the first block of data comprises:

      breaking the previously stored data into a first set of one or more blocks, the first set of one or more blocks including the first block of data;

25       breaking the write data into a second set of one or more blocks, the second set of one or more blocks including the second block of data;

      performing a hash function on the first block of data to obtain a first hash value and on the second block of data to obtain a second hash value, wherein the hash function is deterministic and configured to produce probabilistically unique hash values; and

30       determining that the first hash value is identical to the second hash value by comparing the first hash value to the second hash value.

32. The method of claim 31, wherein the pointer pointing to the first block of data comprises the first hash value or the second hash value.

33. The method of claim 31, wherein the size of the blocks in the first set of one or more blocks and the second set of one or more blocks is either fixed or variable.

34. The method of claim 28, wherein the steps of determining that the second block of data is identical to the first block of data and storing the pointer in the aggregated storage capacity are performed either as the second data is written to the physical storage address or after the second data has already been stored at the physical storage address.

35. A method of pooling storage capacity from a plurality of storage devices and de-duplicating data within the pooled storage capacity, the method comprising:

applying a virtualization layer to a plurality of storage devices to pool storage capacity from the plurality of storage devices, wherein original data is stored in the pooled storage capacity;

receiving a write request from a first host system, the write request including at least a virtual storage address and write data;

mapping the virtual storage address to a physical storage address within a portion of the pooled storage capacity allocated to the host system;

comparing the write data to the original data to identify a first data block within the original data that is identical to a second data block within the write data; and

replacing one of the first data block and the second data block with a pointer that points to a remaining one of the first data block and the second data block in the pooled storage capacity.

36. The method of claim 35, wherein the plurality of storage devices comprise a plurality of heterogeneous storage devices, the abstraction layer presenting a logical representation of the pooled storage capacity to a plurality of host systems including the first host system.

37. The method of claim 35, wherein the virtualization layer is implemented: in the first host system, in a storage controller of at least one of the plurality of storage devices, or in a switch or other appliance of a storage area network to which the plurality of storage devices are coupled.

38. The method of claim 35, further comprising:

receiving a read request from the first host system, the read request including the same virtual storage address as the write request;

mapping the virtual storage address to the physical storage address;

if the second data block within the write data was replaced with a pointer that points to the first data block within the original data, using the pointer to access and return the first data block in response to the read request; and

5 if the first data block was replaced with a pointer that points to the second data block, accessing and returning the second data block in response to the read request.

39. The method of claim 35, wherein comparing the write data to the original data to identify at least a first data block within the original data that is identical to a second data block within the write data comprises:

10 breaking the original data into one or more data blocks including the first data block;

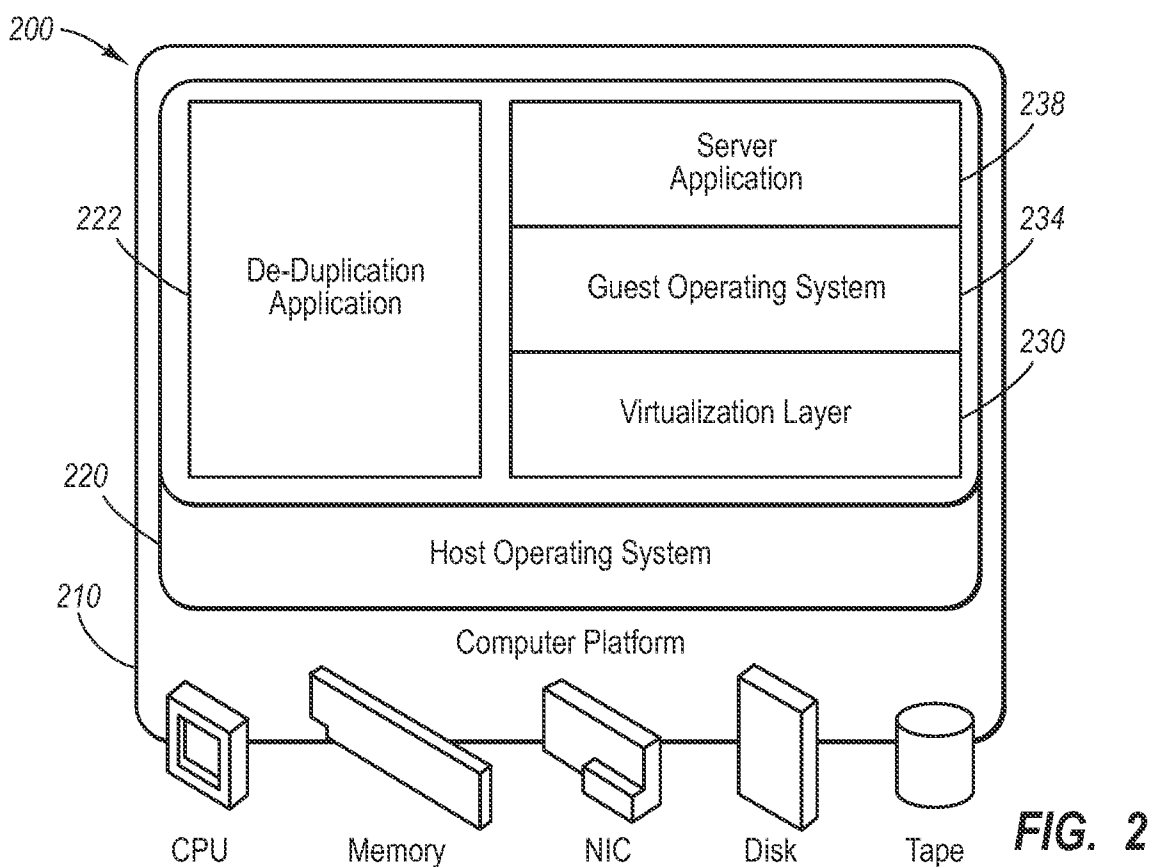
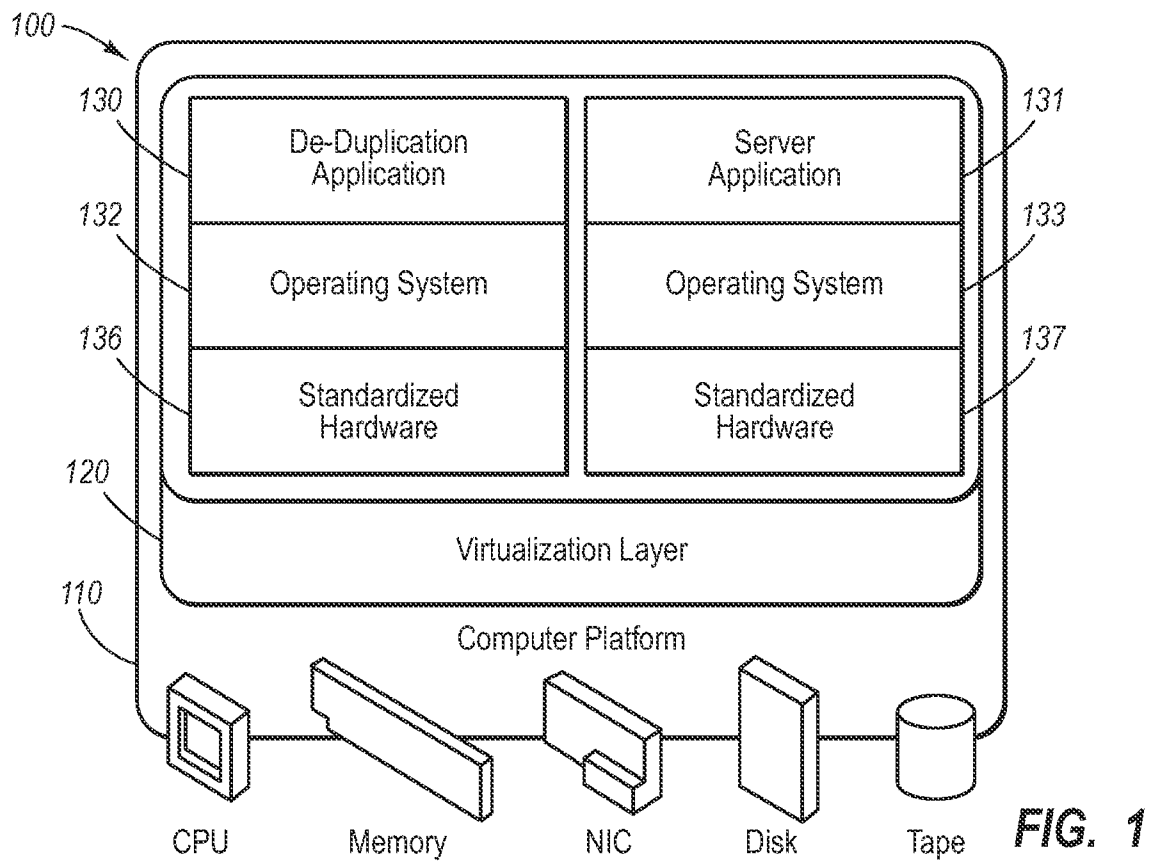
breaking the write data into one or more data blocks including the second data block;

15 hashing the first data block to produce a first hash and the second data block to produce a second hash; and

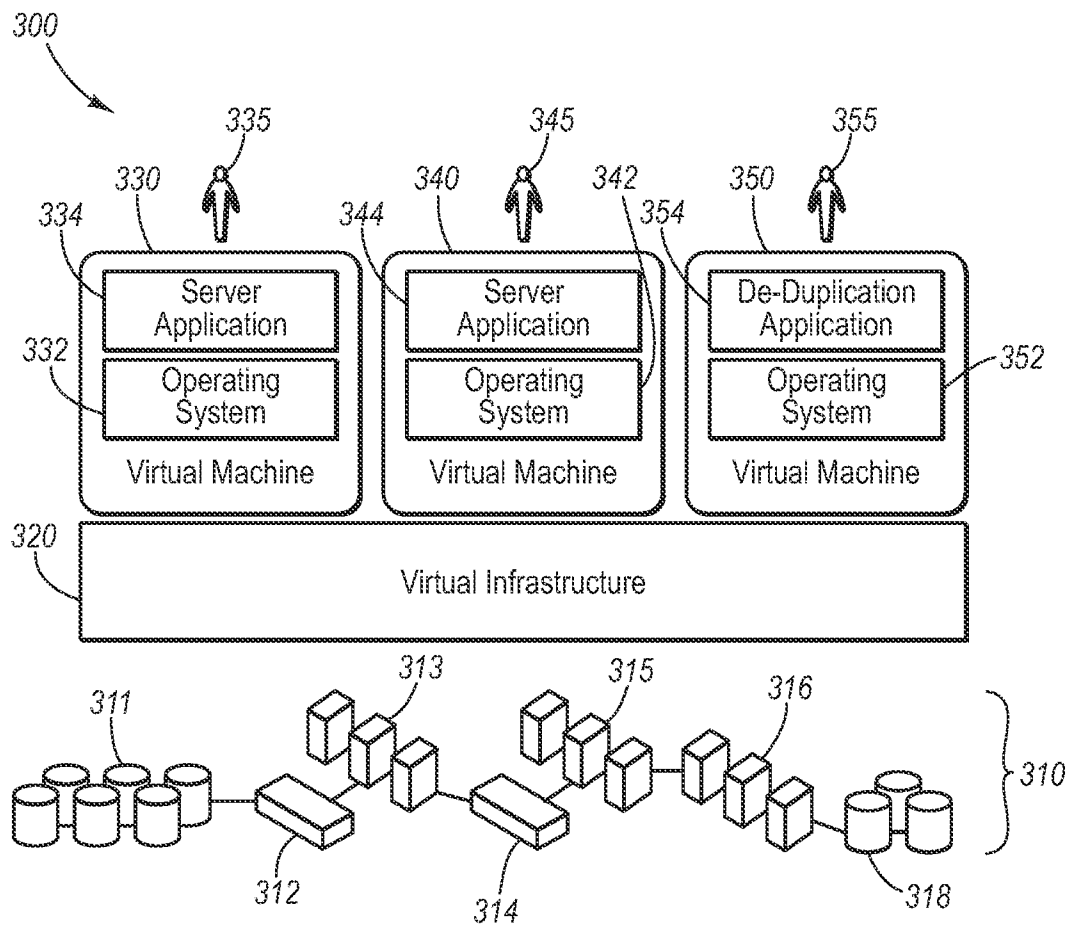
comparing the first hash and the second hash.

40. The method of claim 35, wherein the steps of comparing the write data to the original data and replacing one of the first data block and the second data block with a pointer is performed by a data de-duplication application operating on a second host  
20 system, the virtualization layer presenting a representation of all of the pooled storage capacity to the second host system.

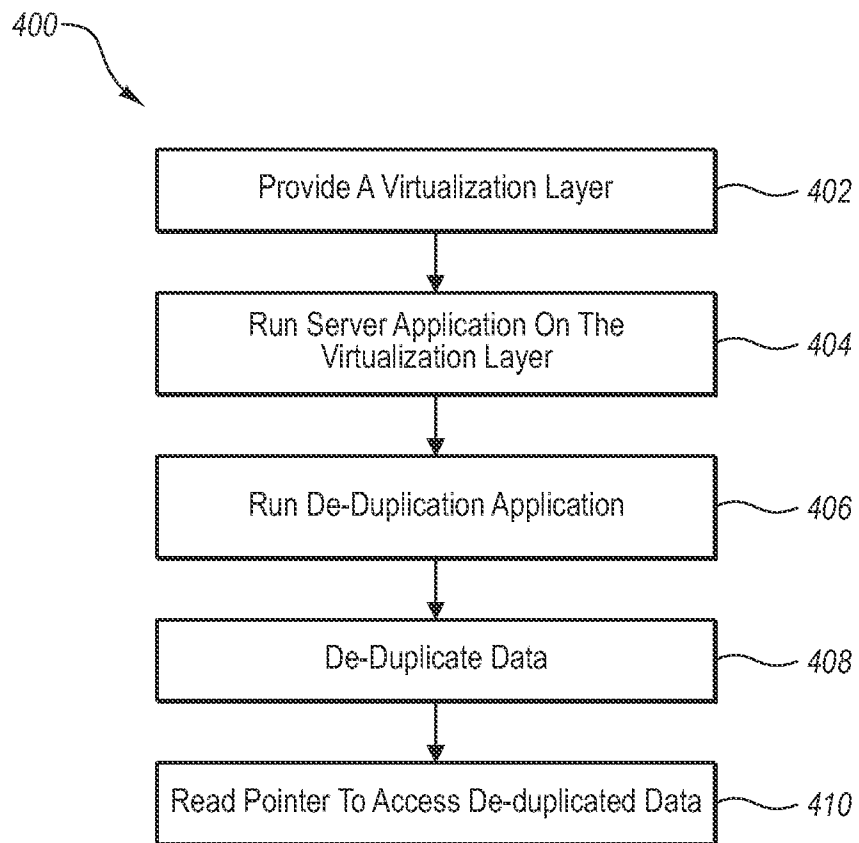
1 / 8



2 / 8

**FIG. 3**

3 / 8

**FIG. 4**

4 / 8

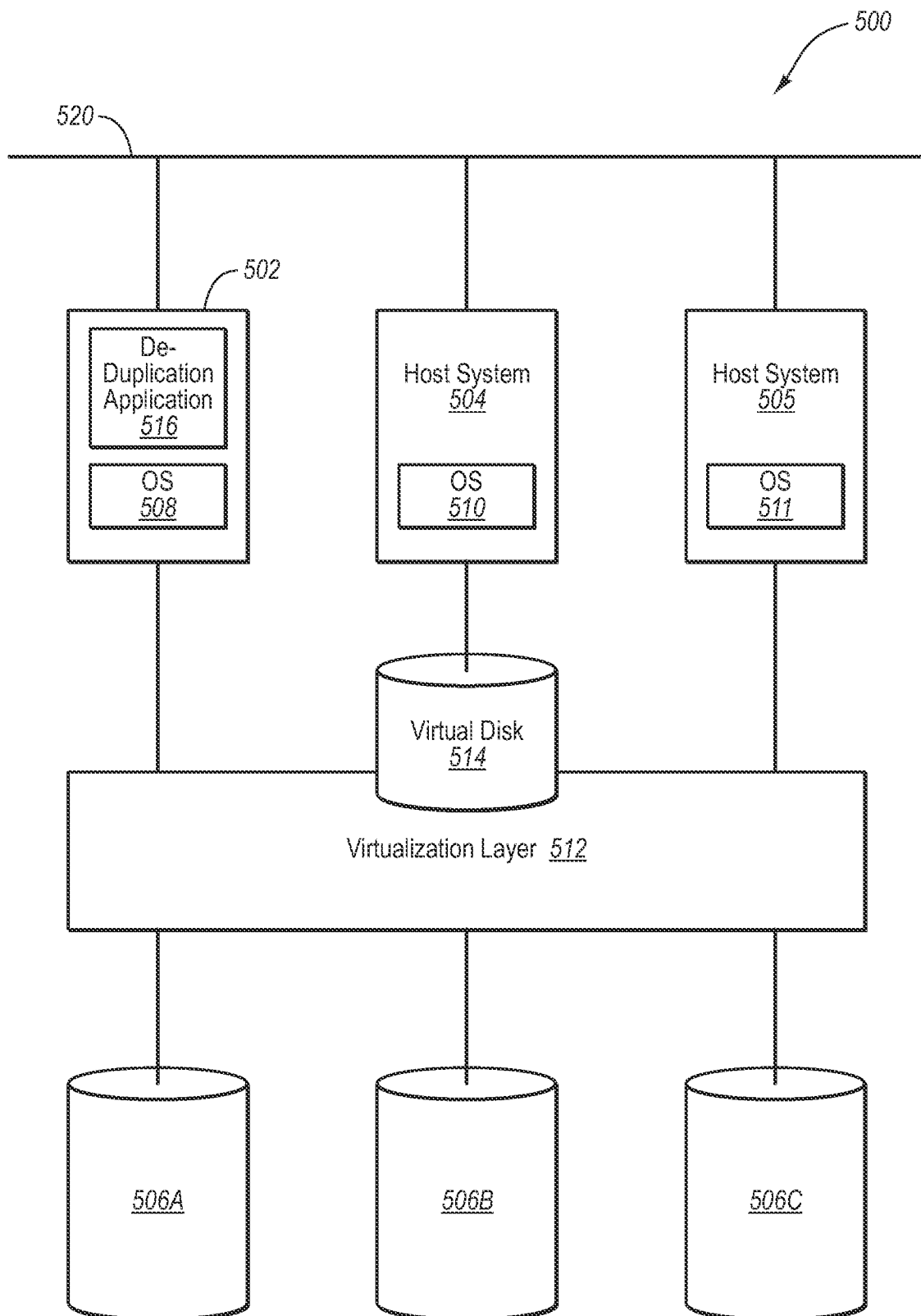


FIG. 5

5 / 8

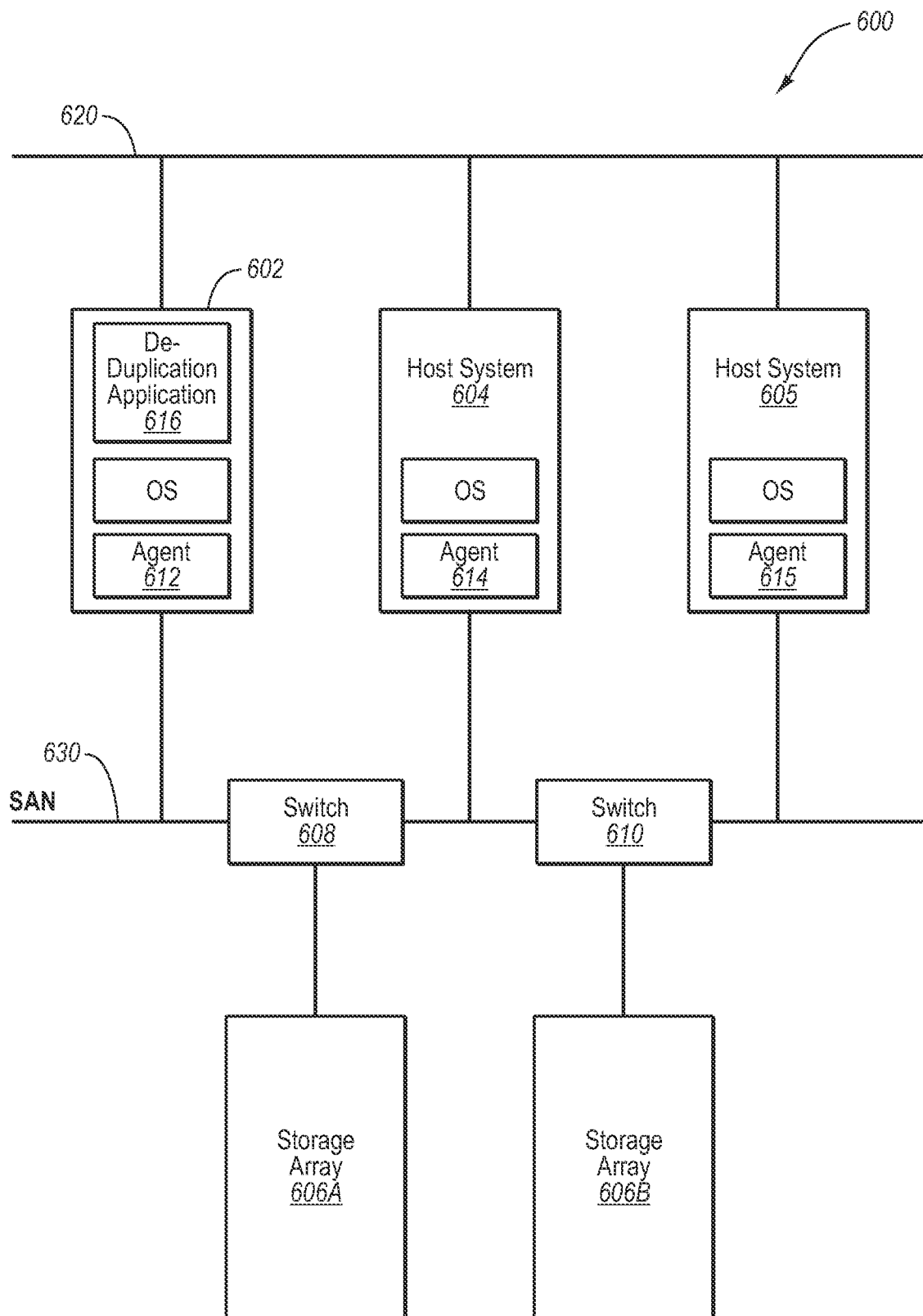


FIG. 6

6 / 8

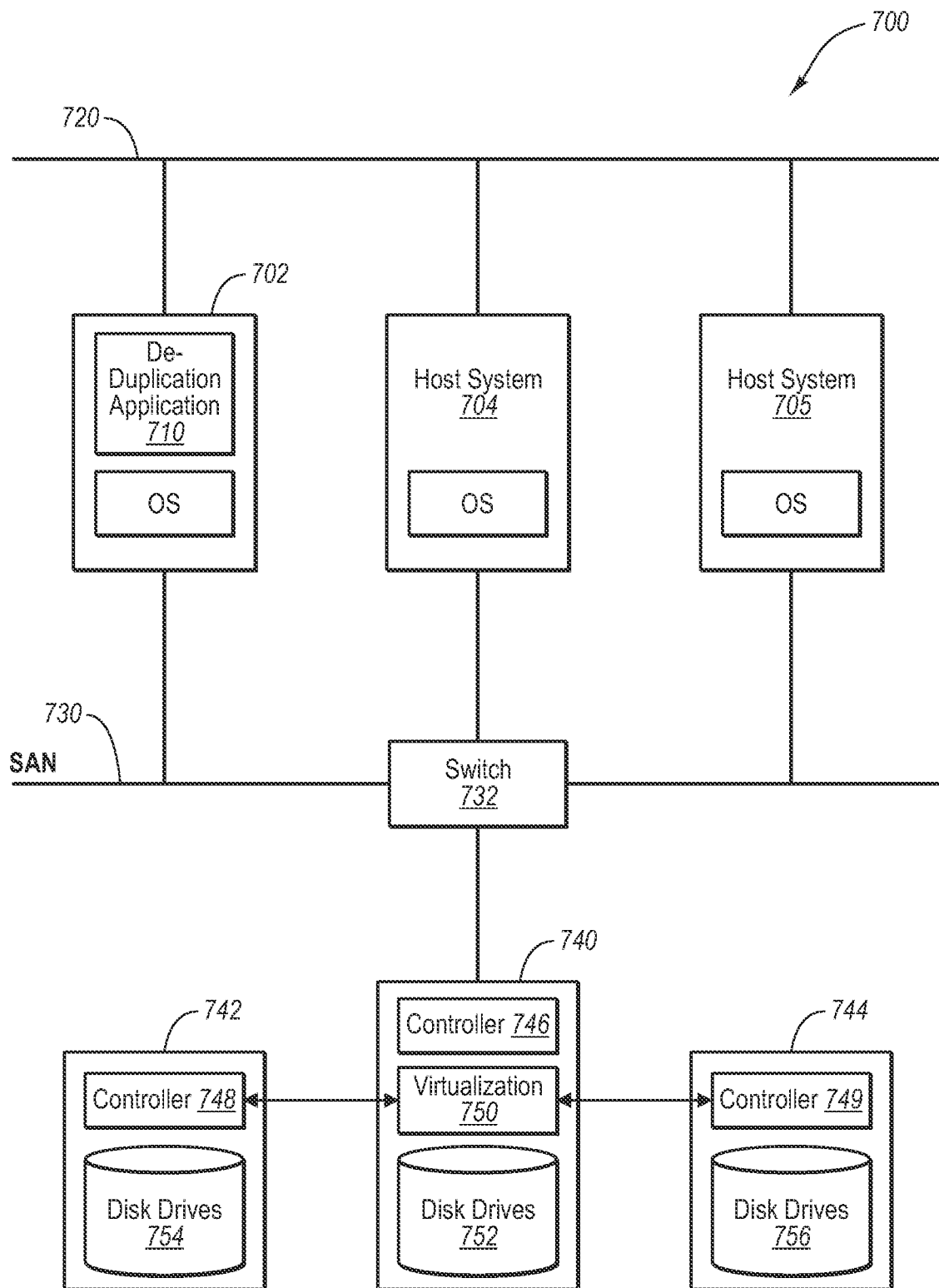


FIG. 7

7/8

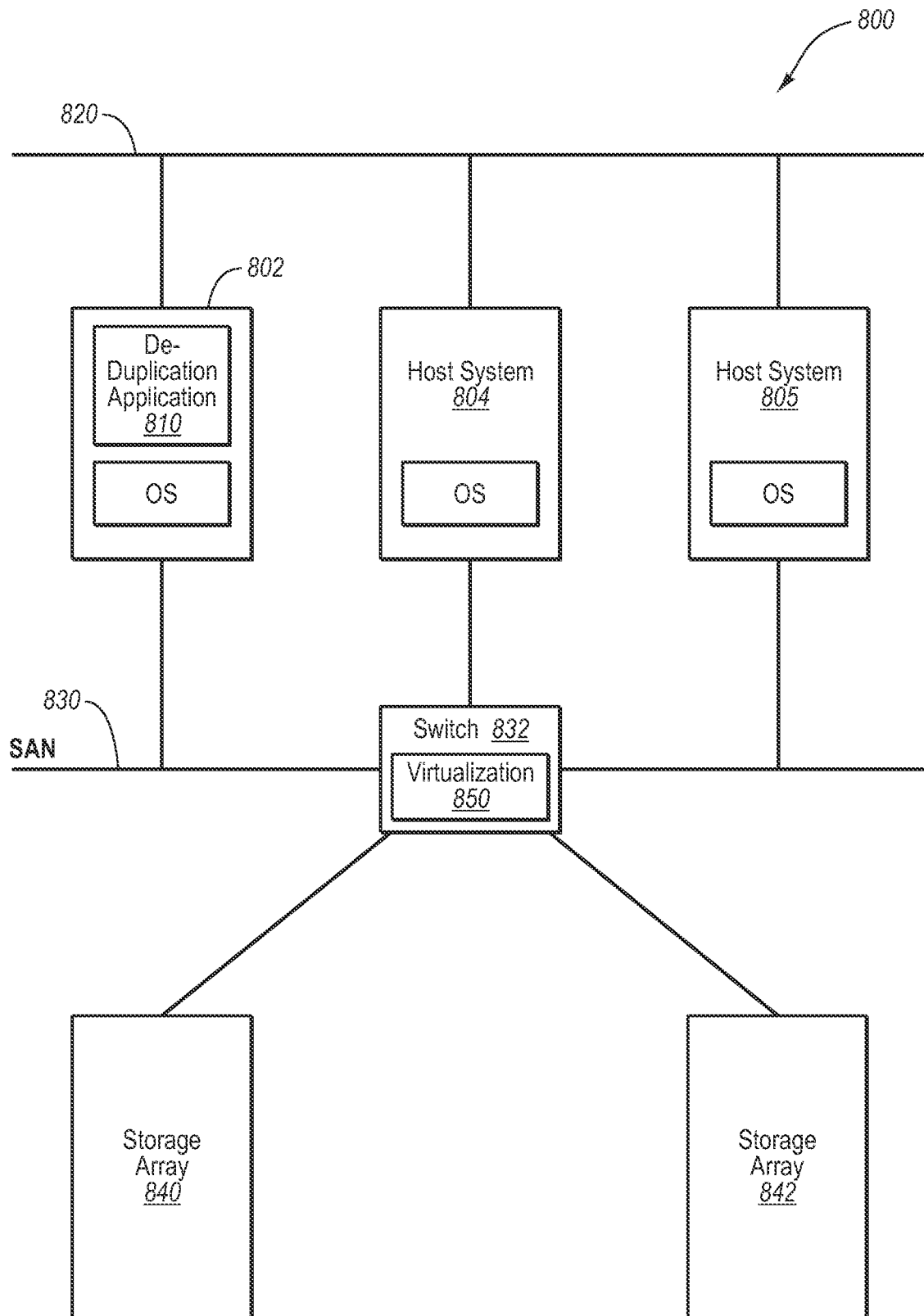
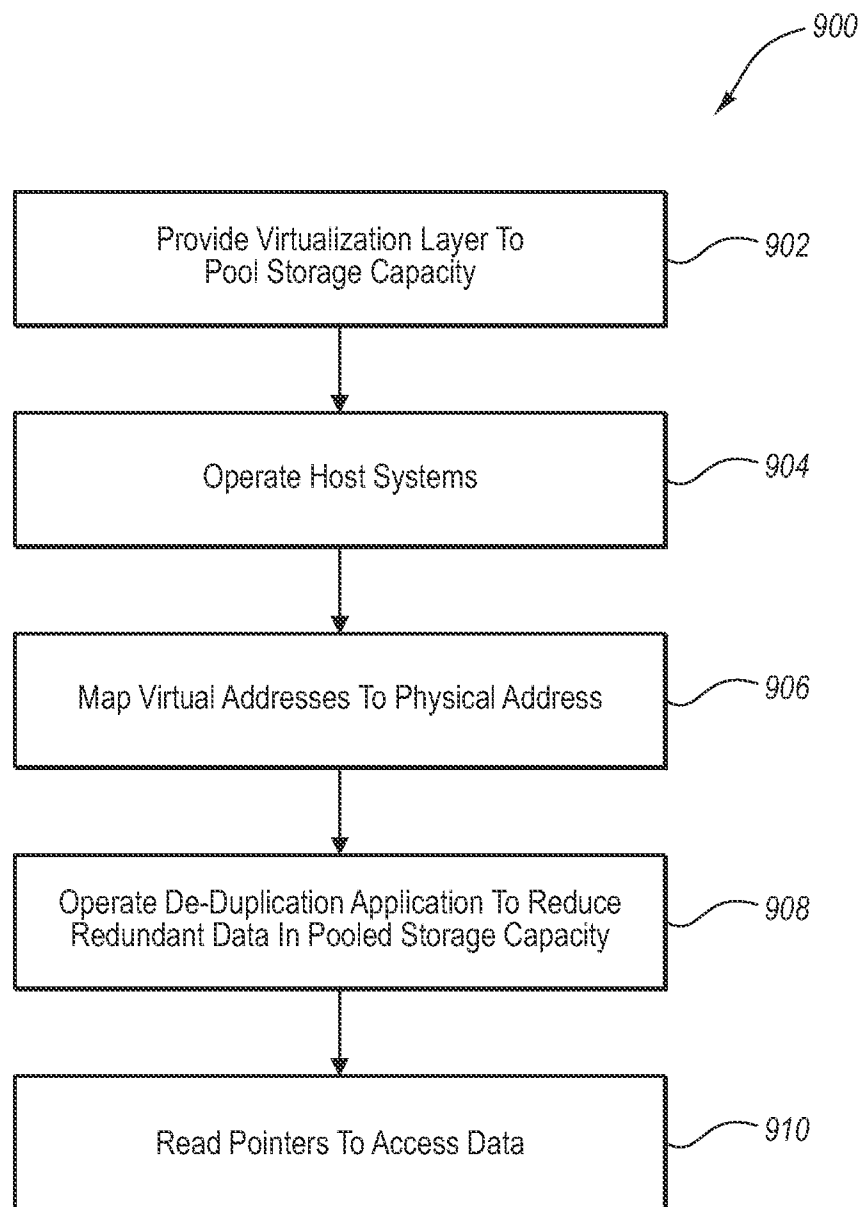


FIG. 8

8 / 8

**FIG. 9**