US 20220043607A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2022/0043607 A1**

MILLER et al. (43) **Pub. Date: Feb. 10, 2022**

(54) **EXPANDING AN ADDRESS SPACE SUPPORTED BY A STORAGE SYSTEM**

(71) Applicant: **PURE STORAGE, INC.**, MOUNTAIN VIEW, CA (US)

(72) Inventors: **ETHAN MILLER**, SANTA CRUZ, CA (US); **JOHN COLGROVE**, LOS ALTOS, CA (US); **JOHN HAYES**, MOUNTAIN VIEW, CA (US)

(21) Appl. No.: **17/508,533**

(22) Filed: **Oct. 22, 2021**

Related U.S. Application Data

(63) Continuation of application No. 16/519,832, filed on Jul. 23, 2019, now Pat. No. 11,169,745, which is a continuation of application No. 14/073,637, filed on Nov. 6, 2013, now Pat. No. 10,365,858.

Publication Classification

(51) **Int. Cl.**

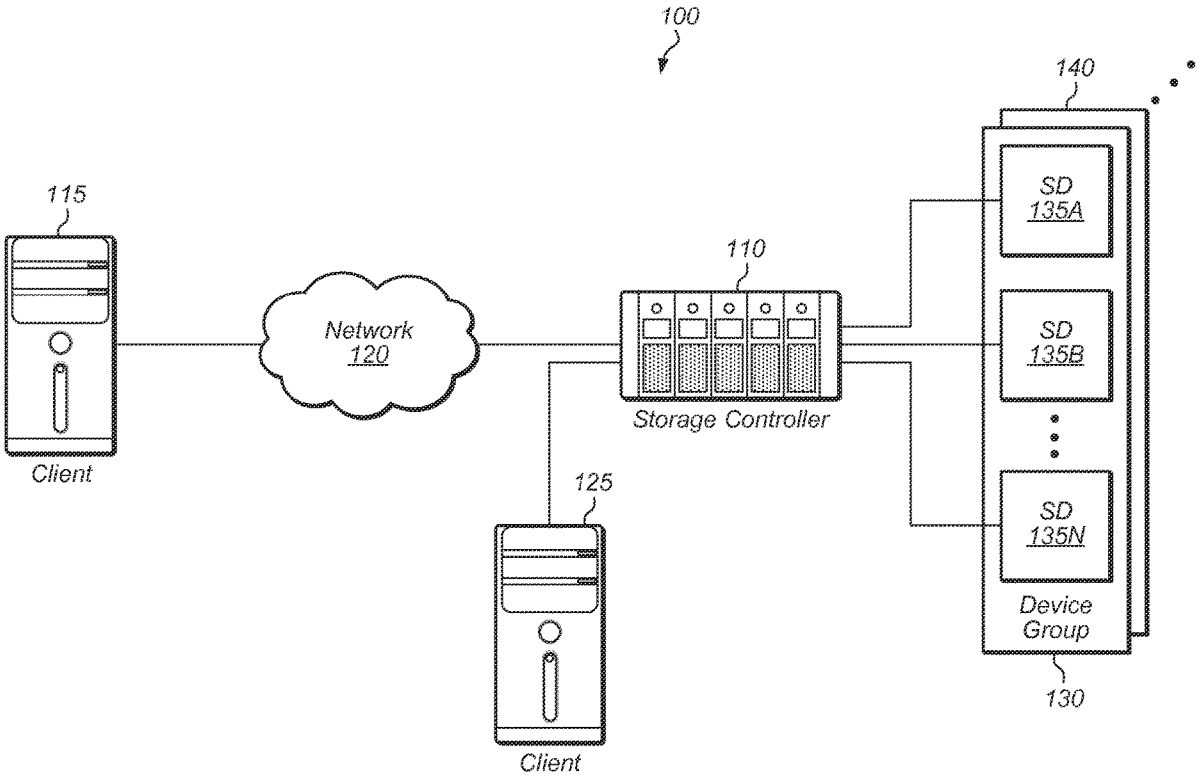
G06F 3/06	(2006.01)
G06F 12/1036	(2006.01)
G06F 12/1009	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 3/0665*** (2013.01); ***G06F 12/1036*** (2013.01); ***G06F 3/0688*** (2013.01); ***G06F 3/0626*** (2013.01); ***G06F 3/064*** (2013.01); ***G06F 12/1009*** (2013.01)

(57) **ABSTRACT**

An apparatus, method, and computer-readable storage medium for allowing a block-addressable storage device to provide a sparse address space to a host computer. The storage device exports an address space to a host computing device which is larger than the storage capacity of the storage device. The storage device translates received file system object addresses in the larger address space to physical locations in the smaller address space of the storage device. This allows the host computing device more flexibility in selecting addresses for file system objects which are stored on the storage device.



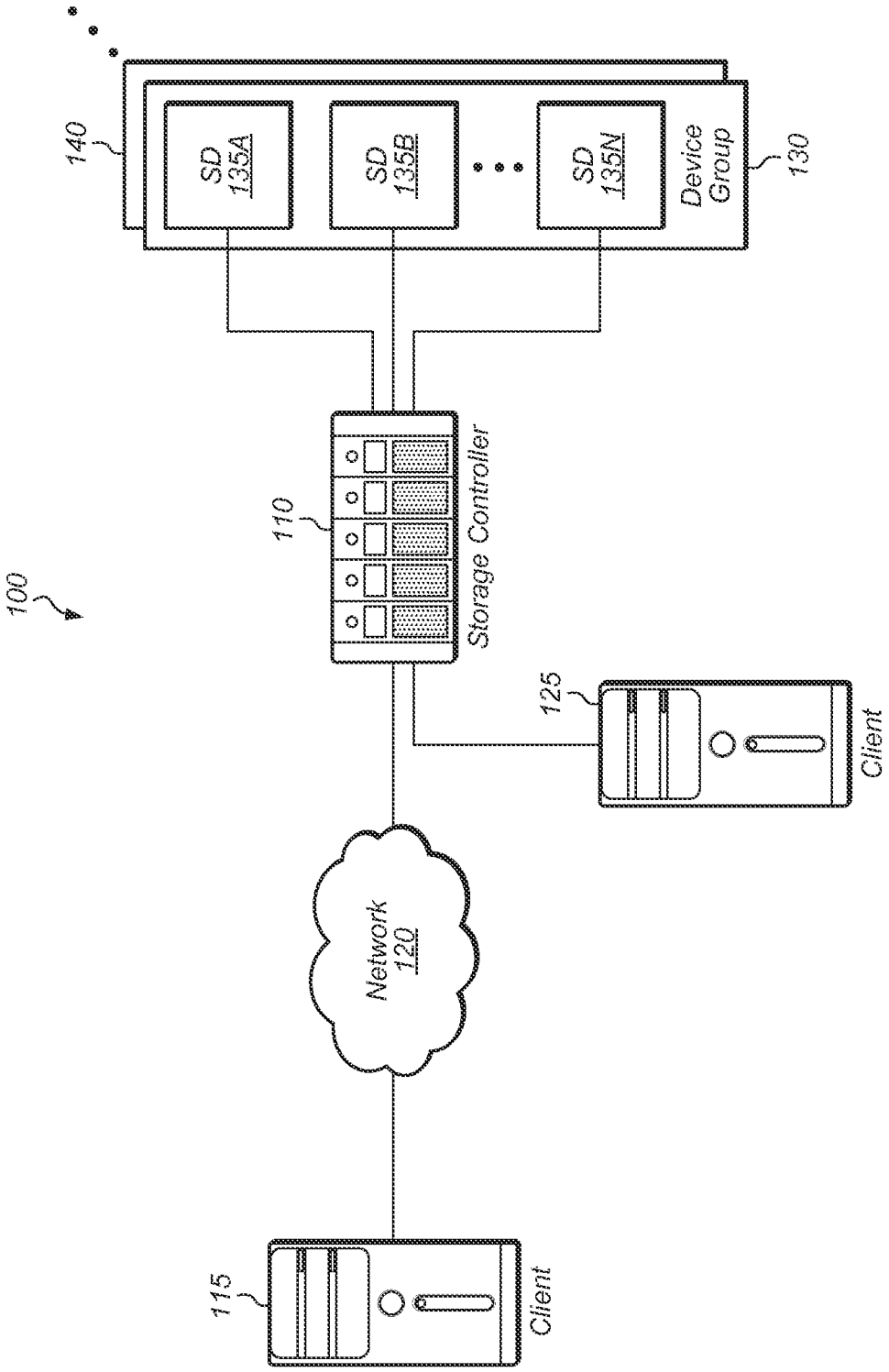

100
110
Storage Controller
140
SD 135A
SD 135B
SD 135N
Device Group
130
125
Client
Network 120
115
Client

FIG. 1

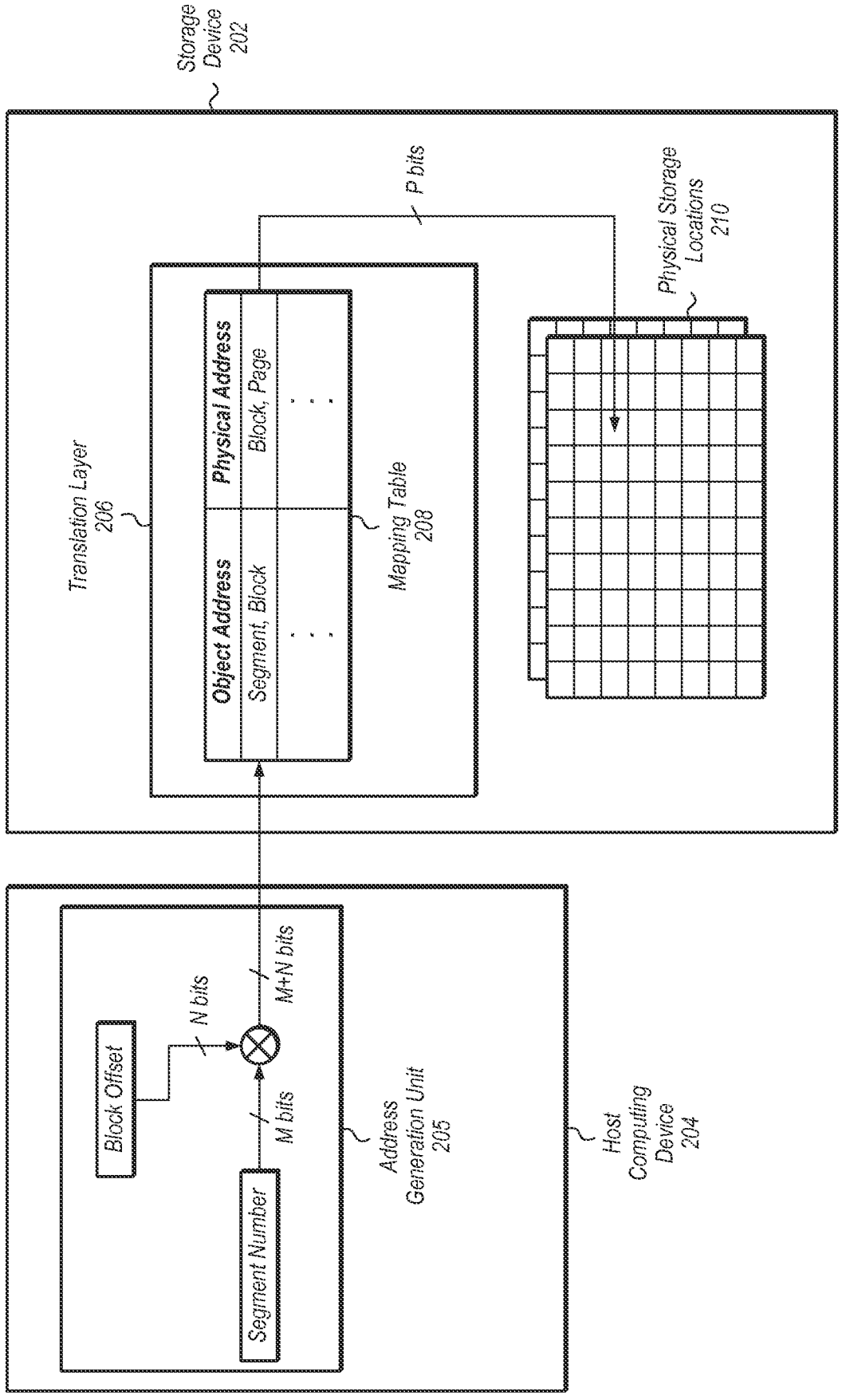

Storage Device 202
Translation Layer 206
Object Address
Segment, Block
Physical Address
Block, Page
Mapping Table 208
P bits
Physical Storage Locations 210
Host Computing Device 204
Address Generation Unit 205
Segment Number
M bits
Block Offset
N bits
M+N bits

FIG. 2

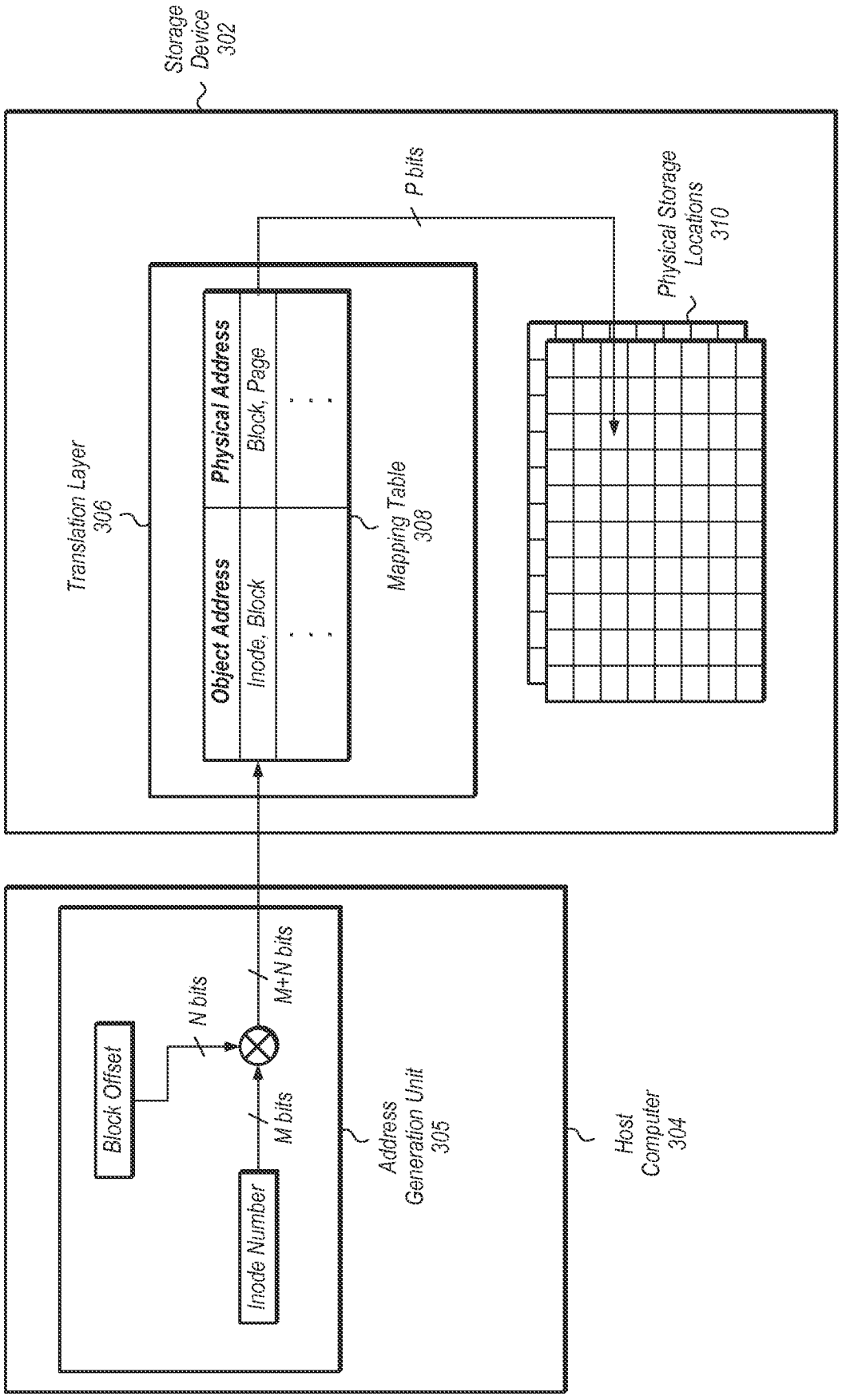

Storage Device 302
Translation Layer 306
Object Address
Physical Address
Inode, Block
Block, Page
Mapping Table 308
P bits
Physical Storage Locations 310
Host Computer 304
Address Generation Unit 305
Block Offset
Inode Number
N bits
M bits
M+N bits

FIG. 3

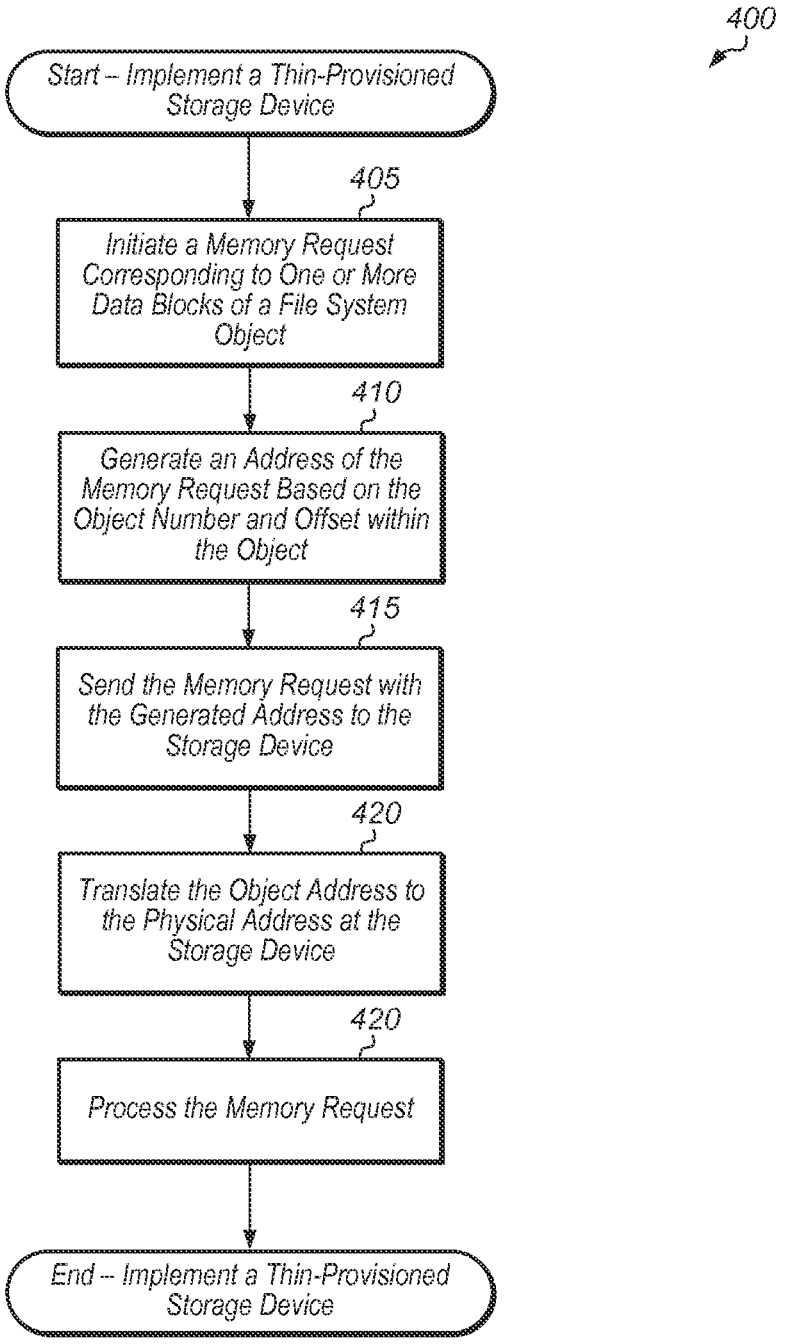

400
Start – Implement a Thin-Provisioned Storage Device
405
Initiate a Memory Request Corresponding to One or More Data Blocks of a File System Object
410
Generate an Address of the Memory Request Based on the Object Number and Offset within the Object
415
Send the Memory Request with the Generated Address to the Storage Device
420
Translate the Object Address to the Physical Address at the Storage Device
420
Process the Memory Request
End – Implement a Thin-Provisioned Storage Device

FIG. 4

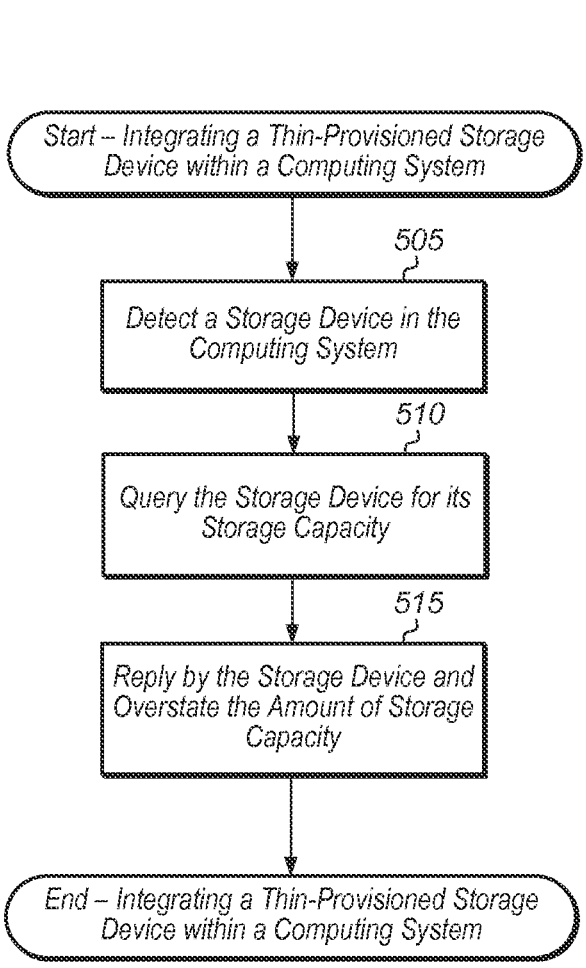

500
Start – Integrating a Thin-Provisioned Storage Device within a Computing System
505
Detect a Storage Device in the Computing System
510
Query the Storage Device for its Storage Capacity
515
Reply by the Storage Device and Overstate the Amount of Storage Capacity
End – Integrating a Thin-Provisioned Storage Device within a Computing System

FIG. 5

EXPANDING AN ADDRESS SPACE SUPPORTED BY A STORAGE SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This is a continuation application for patent entitled to a filing date and claiming the benefit of earlier-filed U.S. Pat. No. 11,169,745, issued Nov. 9, 2021, herein incorporated by reference in its entirety, which is a continuation of U.S. Pat. No. 10,365,858, issued Jul. 30, 2019.

BACKGROUND

Field of the Invention

[0002] This invention relates to storage systems and, more particularly, to techniques for implementing thin provisioning in storage devices.

Description of the Related Art

[0003] The use of solid-state storage devices (e.g., flash memory) in computer systems and storage systems is increasing due to the performance of flash memory as compared to traditional disk storage devices (e.g., hard disk drives (HDDs)). Flash memory offers low-power consumption combined with fast, random (input/output) I/O accesses as compared to traditional disk storage technology. Until recently, flash memory was used primarily in embedded applications, but the use of flash memory has expanded to other applications including desktop and enterprise storage.

[0004] Embedded applications which use flash memory typically include custom operating systems and custom file systems which are designed around the advantages and limitations of flash memory. However, when using flash memory in different applications, such as in desktop and enterprise storage, these systems often have legacy operating systems and file systems which are designed and optimized for use with HDD storage technology. These legacy operating systems and file systems are not able to take full advantage of all of the characteristics of flash memory. Also, these legacy systems may reduce the effective life of flash memory if wear leveling and other techniques are not utilized to prevent frequent writes to the same flash memory locations.

[0005] Additionally, inefficiencies are often introduced when integrating host storage systems with storage devices such as flash memory. For example, host file systems typically need to maintain a mapping between a logical location of a data block, as within a file, and the physical address of that block on the storage device. However, since the physical location of a block may change due to garbage collection and other device-internal processes, flash memory storage devices also maintain a mapping between the block address as provided by the host and the actual physical address of the block. Hence, these approaches require duplication of block mapping in both the file system and on the storage device itself.

[0006] In view of the above, improved systems and methods for providing more efficient interactions between host computing devices and storage devices are desired.

SUMMARY OF EMBODIMENTS

[0007] Various embodiments of apparatuses and methods for implementing thin provisioned storage devices are contemplated.

[0008] In one embodiment, a computer system may include a storage device coupled to a host computer, and the storage device may export a sparse address space to the host computer. The host computer may track file system objects and generate addresses for memory operations based on a file system object identifier (ID) and an offset within the file system object. In one embodiment, host computer may generate an object address for a data block by concatenating an object ID or number with an offset corresponding to the location of the data block within the object. The host computer may convey the object address to the storage device as part of a memory operation for this data block. This scheme simplifies address generation and allows the host computer to avoid having to maintain a mapping table of segments to physical locations.

[0009] The storage device may be configured to maintain mappings between the sparse address space utilized by the host computer and actual physical locations on the storage device. In one embodiment, the storage device may include a mapping table to map an object number and block offset combination to a physical storage location.

[0010] In one embodiment, the host computer may query the storage device to determine the storage capacity of the storage device. In response to receiving this query, the storage device may overstate its capacity to the host computer, responding with an amount of storage which exceeds the actual capacity of the storage device. In one embodiment, the storage device may respond to the query by informing the host computer that the storage device has the maximum possible capacity based on the size of the address field in the interface connection between the host computer and the storage device.

[0011] These and other embodiments will become apparent upon consideration of the following description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. **1** is a generalized block diagram illustrating one embodiment of a storage system.

[0013] FIG. **2** is a generalized block diagram of one embodiment of a host computing device coupled to a storage device.

[0014] FIG. **3** is a generalized block diagram of another embodiment of a host computing device coupled to a storage device.

[0015] FIG. **4** is a generalized flow diagram illustrating one embodiment of a method for implementing a thin-provisioned storage device.

[0016] FIG. **5** is a generalized flow diagram illustrating one embodiment of a method for integrating a thin-provisioned storage device within a computing system.

[0017] While the invention is susceptible to various modifications and alternative forms, specific embodiments are shown by way of example in the drawings and are herein described in detail. It should be understood, however, that drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the invention is to cover all modifications,

equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION

[0018] In the following description, numerous specific details are set forth to provide a thorough understanding of the present invention. However, one having ordinary skill in the art should recognize that the invention might be practiced without these specific details. In some instances, well-known circuits, structures, signals, computer program instruction, and techniques have not been shown in detail to avoid obscuring the present invention. It will be appreciated that for simplicity and clarity of illustration, elements shown in the figures have not necessarily been drawn to scale. For example, the dimensions of some of the elements may be exaggerated relative to other elements.

[0019] This specification includes references to "one embodiment". The appearance of the phrase "in one embodiment" in different contexts does not necessarily refer to the same embodiment. Particular features, structures, or characteristics may be combined in any suitable manner consistent with this disclosure. Furthermore, as used throughout this application, the word "may" is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). Similarly, the words "include", "including", and "includes" mean including, but not limited to.

[0020] Terminology. The following paragraphs provide definitions and/or context for terms found in this disclosure (including the appended claims):

[0021] "Comprising." This term is open-ended. As used in the appended claims, this term does not foreclose additional structure or steps. Consider a claim that recites: "A computing system comprising a plurality of storage devices . . . ." Such a claim does not foreclose the computing system from including additional components (e.g., a network interface, one or more processors, a storage controller).

[0022] "Configured To." Various units, circuits, or other components may be described or claimed as "configured to" perform a task or tasks. In such contexts, "configured to" is used to connote structure by indicating that the units/circuits/components include structure (e.g., circuitry) that performs the task or tasks during operation. As such, the unit/circuit/component can be said to be configured to perform the task even when the specified unit/circuit/component is not currently operational (e.g., is not on). The units/circuits/components used with the "configured to" language include hardware—for example, circuits, memory storing program instructions executable to implement the operation, etc. Reciting that a unit/circuit/component is "configured to" perform one or more tasks is expressly intended not to invoke 35 U.S.C. § 112, sixth paragraph, for that unit/circuit/component. Additionally, "configured to" can include generic structure (e.g., generic circuitry) that is manipulated by software and/or firmware (e.g., an FPGA or a general-purpose processor executing software) to operate in manner that is capable of performing the task(s) at issue. "Configured to" may also include adapting a manufacturing process (e.g., a semiconductor fabrication facility) to fabricate devices (e.g., integrated circuits) that are adapted to implement or perform one or more tasks.

[0023] "Based On." As used herein, this term is used to describe one or more factors that affect a determination. This term does not foreclose additional factors that may affect a determination. That is, a determination may be solely based on those factors or based, at least in part, on those factors. Consider the phrase "determine A based on B." While B may be a factor that affects the determination of A, such a phrase does not foreclose the determination of A from also being based on C. In other instances, A may be determined based solely on B.

[0024] Referring now to FIG. **1**, a generalized block diagram of one embodiment of a storage system **100** is shown. Storage system **100** may include storage controller **110** and storage device groups **130** and **140**, which are representative of any number of storage device groups (or data storage arrays). As shown, storage device group **130** includes storage devices **135**A-N, which are representative of any number and type of storage devices (e.g., solid-state drives (SSDs)). Storage device group **140** may also include a plurality of storage devices which are not shown to avoid obscuring the figure.

[0025] Storage controller **110** may be coupled directly to client computer system **125**, and storage controller **110** may be coupled remotely over network **120** to client computer system **115**. Clients **115** and **125** are representative of any number of clients which may utilize storage controller **110** for storing and accessing data in system **100**. It is noted that some systems may include only a single client, connected directly or remotely to storage controller **110**.

[0026] Storage controller **110** may include software and/or hardware configured to provide access to storage devices **135**A-N. Although storage controller **110** is shown as being separate from storage device groups **130** and **140**, in some embodiments, portions or the entirety of storage controller **110** may be located within one or each of storage device groups **130** and **140**. Storage controller **110** may include or be coupled to a base operating system (OS), a volume manager, and additional control logic for implementing the various techniques disclosed herein.

[0027] Storage controller **110** may include and/or execute on any number of processors and may include and/or execute on a single host computing device or be spread across multiple host computing devices, depending on the embodiment. The host computing device(s) may be servers, workstations, or other types of computing devices. In some embodiments, storage controller **110** may include or execute on one or more file servers and/or block servers. Storage controller **110** may use any of various techniques for replicating data across devices **135**A-N to prevent loss of data due to the failure of a device or the failure of storage locations within a device.

[0028] The file system of storage controller **110** may manage blocks of data according to a file system object, such as a file, volume, segment, or other type of object. Each block of data may be associated with the identifier of the object (e.g., file ID, segment number) and the offset of the block within the object. In one embodiment, storage controller **110** may provide a given storage device with file system object information directly, allowing the given storage device to maintain the mapping between the file system object and the actual physical location, and freeing storage controller **110** from having to maintain this mapping information. Instead, storage controller **110** may maintain only a minimal amount of information about the file system objects. For example, in one embodiment, storage controller **110** may maintain a table with one entry per object detailing

information including the object's creation time, current usage statistics, and an object ID. Accordingly, block address generation may be simplified for storage controller **110** without needing to reference a complex mapping table to determine the address of a specific block of a given object. When storage controller **110** creates a new object, the new object may be given an ID one greater than any previously-used object ID. In this way, storage controller **110** can ensure that the blocks in this new object do not reuse any previously used addresses.

[0029] In one embodiment, a given storage device **135** may receive read and write memory operations generated by storage controller **110**. Each memory operation may include an address composed of a file system object number (or ID) and an offset for the data block in question. Each storage device may include a translation layer, and the translation layer may include a mapping table to map an object number and offset received with a given memory operation to a corresponding physical storage location. The translation layer may export an address space larger than the capacity of the storage device, which allows storage controller **110** to write a data block to a first address, wherein the first address has a value larger than a capacity of block storage locations in the storage device.

[0030] For example, in one embodiment, a given storage device **135**A may have 1024 addressable storage locations for storing a block of data. The given storage device **135**A may utilize 10 address bits to address these 1024 storage locations. The translation layer on this given storage device **135**A may allow storage controller **110** to write to addresses with more than 10 address bits. For example, storage controller **110** may utilize 16 address bits for addressing memory operations to the given storage device **135**A. This provides storage controller **110** with 65,536 different block addresses to use for memory operations targeting the given storage device **135**A. The storage controller **110** may actually only use a small percentage of these 65,536 addressable locations to prevent the given storage device **135**A from being oversubscribed. Therefore, the address space of storage controller **110** may be described as being a sparse address space, where only a portion of the total address space is being utilized at any given time. The translation layer of the given storage device **135**A may map received 16-bit length addresses to 10-bit addresses corresponding to actual physical storage locations. It is noted that this scenario of using a 16-bit address length at storage controller **110** and a 10-bit address length at storage device **135**A is merely one example of a thin-provisioned storage device implementation. Other bit-lengths for addresses at the storage controller **110** and the storage device **135**A may be utilized in other embodiments. Generally speaking, the address space size may be larger at storage controller **110** than the total amount of free space on storage devices **135**A. In some embodiments, the bit-lengths for addresses at storage controller **110** and storage devices **135**A may be the same size, but there may be more available storage locations at storage controller **110** than in storage devices **135**A.

[0031] In some embodiments, storage controller **110** may be coupled to multiple storage devices **135**A-N. The total number of addressable storage locations on all of the storage devices coupled to storage controller **110** may be less than the number of addresses in the address space utilized by storage controller **110**. For example, in one scenario, storage controller **110** may be coupled to 16 storage devices, and each storage device may have 2^16 addressable storage locations for data blocks. The combined total of all addressable storage locations on all 16 storage devices in this scenario would be 2^20 (2^16*16 storage devices). The storage controller **110** may utilize an address space of 2^48 addresses in this scenario, which exceeds the combined capacity of all 16 storage devices. Other scenarios may have other numbers of storage devices with other capacities, and the combined total capacity of all storage devices coupled to the storage controller may be less than the size of the address space utilized by the storage controller for generating object addresses.

[0032] In embodiments where storage controller **110** is coupled to multiple storage devices, storage controller **110** may utilize any of various techniques for spreading memory operations out to the multiple storage devices. For example, in one embodiment, storage controller **110** may determine which storage device to send a given memory operation based on a modulo operation performed on the offset portion of the address. Accordingly, in an embodiment with 8 storage devices, storage controller **110** may take the offset portion of the address modulo 8 and use this to map a value of 0 to the first storage device, a value of 1 to the second storage device, and so on. Other techniques for distributing memory operations to multiple storage devices are possible and are contemplated. These techniques may also be utilized for other types of computing devices (besides storage controllers) which are coupled to multiple storage devices.

[0033] In one embodiment, a given storage device **135**B may declare itself to have a larger capacity than it actually possesses. Storage controller **110** may receive and believe this "dishonest" declaration from the given storage device **135**B. Accordingly, storage controller **110** may operate under the assumption that each block address being generated by its file system and conveyed to the given storage device **135**B is being mapped directly to an addressable block storage location. In effect, the given storage device **135**B is hiding the fact that it is thin-provisioned from storage controller **110**, and device **135**B is also hiding the underlying mapping of object addresses to physical addresses from storage controller **110**. This scheme may allow legacy operation systems, file systems, interface protocols, and/or interface logic to be used by storage controller **110** when interfacing with solid-state storage devices. The given storage device **135**B may also be able to use legacy interface logic to communicate with storage controller **110** based on using this "dishonest" declaration scheme.

[0034] It is noted that in alternative embodiments, the number and type of client computers, storage controllers, networks, storage device groups, and data storage devices is not limited to those shown in FIG. **1**. At various times one or more clients may operate offline. In addition, during operation, individual client computer connection types may change as users connect, disconnect, and reconnect to system **100**. Further, the systems and methods described herein may be applied to directly attached storage systems or network attached storage systems and may include a host operating system configured to perform one or more aspects of the described methods. Numerous such alternatives are possible and are contemplated.

[0035] Network **120** may utilize a variety of techniques including wireless connection, direct local area network (LAN) connections, wide area network (WAN) connections such as the Internet, a router, storage area network, Ethernet,

and others. Network **120** may comprise one or more LANs that may also be wireless. Network **120** may further include remote direct memory access (RDMA) hardware and/or software, transmission control protocol/internet protocol (TCP/IP) hardware and/or software, router, repeaters, switches, grids, and/or others. Protocols such as Fibre Channel, Fibre Channel over Ethernet (FCoE), iSCSI, and so forth may be used in network **120**. The network **120** may interface with a set of communications protocols used for the Internet such as the Transmission Control Protocol (TCP) and the Internet Protocol (IP), or TCP/IP.

[0036] Client computer systems **115** and **125** are representative of any number of stationary or mobile computers such as desktop personal computers (PCs), servers, server farms, workstations, laptops, handheld computers, servers, personal digital assistants (PDAs), smart phones, and so forth. Generally speaking, client computer systems **115** and **125** include one or more processors comprising one or more processor cores. Each processor core includes circuitry for executing instructions according to a predefined general-purpose instruction set. For example, the x86 instruction set architecture may be selected. Alternatively, the ARM®, Alpha®, PowerPC®, SPARC®, or any other general-purpose instruction set architecture may be selected. The processor cores may access cache memory subsystems for data and computer program instructions. The cache subsystems may be coupled to a memory hierarchy comprising random access memory (RAM) and a storage device.

[0037] Turning now to FIG. **2**, a block diagram of one embodiment of a host computing device coupled to a storage device is shown. Host computing device **204** may include address generation unit **205** for generating addresses for segments of memory operations being conveyed to storage device **202**. Address generation unit **205** may generate addresses for a memory operation from the corresponding segment number and block offset. For example, in one embodiment, segments may be 8 megabytes (MB) in length and blocks may be 4 kilobytes (KB) long, resulting in 2048 blocks per segment. Address generation unit **205** may construct block addresses by concatenating the segment number with an 11-bit block offset and passing the resultant address to storage device **202**. It is noted that the block diagram of address generation unit **205** is a logical representation of address generation unit **205**. Address generation unit **205** may be implemented using hardware and/or software, depending on the embodiment.

[0038] Storage device **202** may include translation layer **206** and non-volatile physical storage locations **210**. Storage device **202** may also include other components (e.g., buffer, processor) which are not shown to avoid obscuring the figure. Translation layer **206** may be part of the interface logic utilized by storage device **202** to interface with host computing device **204** and to process received memory operations. Translation layer **206** may be referred to as a flash translation layer in embodiments where storage device **202** is a flash memory device.

[0039] Translation layer **206** may be configured to translate addresses from a sparse address space utilized by host computing device **204** to the physical address space corresponding to physical storage locations **210**. Translation layer **206** may present an address space to host computing device **204** which is larger than the storage capacity of storage device **202**. Translation layer **206** may be implemented using any suitable combination of hardware and/or software.

[0040] In one embodiment, the interface between host computing device **204** and storage device **202** may be a custom designed interface. Alternatively, in other embodiments, the interface may utilize a standard communication protocol. For example, the interface between host computing device **204** and storage device **202** may utilize a Serial Advanced Technology Attachment ("SATA") protocol bus, a Small Computer System Interface ("SCSI") protocol bus, a Serial Attached SCSI ("SAS") protocol bus, or any of various other communication protocols. In one embodiment, the protocol utilized on the interface between host computing device **204** and storage device **202** may utilize a fixed-width address for addressing blocks. The protocol may also support only a single volume per storage device **202**, and the number of blocks storage device **202** can store may be less than the number of blocks the protocol can address.

[0041] Translation layer **206** may include mapping table **208** for mapping file system object addresses received from host computing device **204** to specific locations within physical storage locations **210**. Mapping table **208** may include multiple levels and may be arranged in any suitable fashion, depending on the embodiment. For example, mapping table **208** may implement a fully associative mapping, set associative mapping, or direct mapping, depending on the embodiment. It is noted that translation layer **206** may also include other logic which is not shown to avoid obscuring the figure.

[0042] In one embodiment, for a given data block managed by host computing device **204**, the segment number of the segment containing the data block may be concatenated with the offset of the data block within the segment to generate the address of the data block, and this address may be conveyed to storage device **202**. The segment number may be M bits and the block offset may be N bits, resulting in an address length of M+N bits which is conveyed to storage device **202**. This approach simplifies address generation and allows host computing device **204** to avoid having to maintain a mapping table to translate addresses of data blocks to logical or physical addresses. Instead, host computing device **204** and storage device **202** are able to share a single mapping table **208** for mapping segments to physical addresses.

[0043] Memory operations may be conveyed by host computing device **204** to storage device **202**, and each memory operation may include an object address generated from the corresponding segment number and block offset. When a memory operation is received by storage device **202**, the segment number and block offset address may be translated to a physical address using mapping table **208**. In one embodiment, each physical address may indicate a corresponding block and page of physical storage locations **210**. The physical address may be of size P bits, wherein P is less than the size (M+N) of the object address received by storage device **202**. In other words, the address space utilized by host computing device **204** to manage segments targeting storage device **202** is larger than the capacity of physical storage in storage device **202**. Accordingly, translation layer **206** may be configured to allow host computing device **204** to write a data block to a first address (in the address space of host computing device **204**), wherein the first address has a value larger than the capacity of block storage locations in storage device **202**. The values P, M, and N are assumed to be positive integers for the purposes of this discussion.

[0044] In one embodiment, with a segment length of 8 MB, if storage device **202** has a capacity of 50,000 segments, then storage device **202** has a storage capacity of 400 gigabytes (GB). However, some of the data utilized by host computing device **204** might not be within the first 400 GB of address space. For example, host computing device **204** might create 200,000 segments interspersed with invalidation of 160,000 segments, leading to a maximum address space of 1.6 terabytes (TB), but only actually storing 320 GB of data on storage device **202**, which is within its 400 GB capacity. If a segment is no longer needed, host computing device **204** may use a TRIM command to inform storage device **202** that it can free the blocks associated with the segment. If a new segment is created, rather than reusing a segment ID from an old segment, host computing device **204** can simply create a new segment ID by incrementing the largest previously-used segment ID.

[0045] If the memory operation is a read operation and a lookup corresponding to the memory operation hits an existing entry in mapping table **208**, then the read operation may be performed to the corresponding physical address retrieved from the hit entry. Write operations which are received by storage device **202** may result in new entries being generated for mapping table **208**. If a write operation hits in mapping table **208**, then the physical address of the hit entry may be invalidated (and erased at a later point in time), and a new entry may be recorded in mapping table **208** which maps the segment number and block offset address to the new physical address allocated for the write operation. During garbage collection operations or other internal-device processes, mapping table **208** may be updated as data is moved between physical storage locations.

[0046] Turning now to FIG. **3**, a block diagram of another embodiment of a host computing device coupled to a storage device is shown. As shown in FIG. **3**, host computer **304** is coupled to storage device **302**. Host computer **304** includes address generation unit **305**, and storage device includes physical storage locations **310** and translation layer **306**, which includes mapping table **308**.

[0047] In contrast to FIG. **2**, address generation unit **305** of FIG. **3** may generate an address of a memory operation based on an inode number instead of using a segment number. Host computer **304** may include a file system which tracks data based on files and block offsets within the file, and the file system may map each file name to an inode number.

[0048] It is noted that in other embodiments, other file system objects may be utilized by address generation unit **305** of host computer **304** for generating addresses associated with memory operations. For example, in another embodiment, the file system may track data according to volumes, and the volume numbers and offsets may be utilized for generating addresses in address generation unit.

[0049] Mapping table **308** of translation layer **306** may map object addresses to physical addresses, and each object address in this case may be an inode number concatenated with an offset. Storage device **302** may be configured in a similar manner to storage device **202** of FIG. **2**, and the description of storage device **202** may also apply to storage device **302**.

[0050] Turning now to FIG. **4**, one embodiment of a method **400** for implementing a thin-provisioned storage device is shown. For purposes of discussion, the steps in this embodiment are shown in sequential order. It should be noted that in various embodiments of the method described below, one or more of the elements described may be performed concurrently, in a different order than shown, or may be omitted entirely. Other additional elements may also be performed as desired.

[0051] A host computing device may initiate a memory request corresponding to one or more data blocks of a file system object (block **405**). The file system object may be a file, segment, volume, or another type of object depending on the embodiment. The host computing device may generate an address of the memory request based on an object number (or ID) and an offset of the first block (of the one or more blocks) within the object (block **410**). The address of the memory request may be N bits in length. Therefore, for an address of N bits in length, there are a total of 2N possible addresses that may be utilized in this address space. In one embodiment, the file system object may be a file and the file system may use the inode number corresponding to the file as the first part of the address. The second part of the address may be the offset of the data block within the file. In one embodiment, the inode number may be represented by 24 bits and the offset may be represented by 24 bits, making the length of the address equal to 48 bits. This would give the file system a total of 248 different addresses in the address space when mapping requests to a storage device coupled to the host computing device. It may be assumed for the purposes of this discussion that the host computing device is coupled to a single storage device.

[0052] Next, the host computing device may send the memory request with the generated address to the storage device (block **415**). In response to receiving the memory request, the storage device may translate the object address of the memory request to a physical address (block **420**). In one embodiment, the storage device may maintain a mapping table to translate received object addresses to physical addresses. If the object address of the memory request is a new address, or if the memory request is a write request, then the storage device may generate a new physical address for this object address and store this new mapping in the mapping table. In one embodiment, the request sent from the host computing device to the storage device may be a TRIM command, and the TRIM command may specify a file system object which has been deleted. A TRIM command is an example of one technique in which the host computing device may inform the storage device that an address range has been deleted and that the storage device may "forget" about a range of identifiers. In response to receiving the TRIM command, the storage device may reclaim the storage locations utilized by these range of identifiers.

[0053] The address space of all possible object addresses may exceed the physical capacity of the storage device. In other words, the bit length of object addresses may be larger than the bit length of physical addresses stored in the table. For example, in one embodiment, object addresses may be 48 bits in length and physical addresses may be 20 bits in length. In other embodiments, other object address bit-lengths and other physical address bit-lengths may be utilized.

[0054] After block **420**, the storage device may process the memory request using the physical address translated from the object address (block **425**). The processing of the memory request may be performed using any of various techniques well known to those skilled in the art. If the

memory request is a TRIM command, the storage controller may reclaim the blocks corresponding to the specified address range. After block **425**, method **400** may end.

[0055] Referring now to FIG. **5**, one embodiment of a method for integrating a thin-provisioned storage device within a computing system is shown. For purposes of discussion, the steps in this embodiment are shown in sequential order. It should be noted that in various embodiments of the method described below, one or more of the elements described may be performed concurrently, in a different order than shown, or may be omitted entirely. Other additional elements may also be performed as desired.

[0056] A computing system may detect a storage device connected to the computing system (block **505**). In one embodiment, the computing system may be a server and the storage device may be a storage device (e.g., solid state drive) in the server. In another embodiment, the computing system may be a tablet and the storage device may be a flash memory device integrated within the tablet. In other embodiments, other types of computing systems (e.g., desktop computer, smartphone) may be utilized with any of various types of storage devices. In response to detecting the presence of the storage device, the computing system may query the storage device for its storage capacity (block **510**). In response to receiving the query, the storage device may "lie" and overstate the amount of storage capacity it contains (block **515**).

[0057] In one embodiment, the storage device may reply to the computing system's query with the maximum possible storage capacity based on the number of supported address bits which are available on the interface connection between the computing system and the storage device. For example, if the storage device uses the Serial Advance Technology Attachment (SATA) protocol to communicate with the computing system host, the number of supported address bits may be 48, and the storage device may reply that it contains 248 addressable blocks, even though this may far exceed the amount of addressable blocks on the storage device. In other embodiments, the storage device may utilize other types of protocols or interfaces to communicate with the computing system, and these other types of interfaces may support other numbers of address bits besides **48**.

[0058] By overstating its storage capacity, the storage device provides the computing system host with more flexibility in selecting addresses for file system objects that are stored on the storage device. For example, when the computing system needs to generate an ID for a new file system object, instead of reusing an ID from an old object, the computing system may increment the largest previously used ID and to create a new ID for the new object. The large address space exported by the storage device enables this flexibility in generating IDs for the computing system.

[0059] It is noted that the above-described embodiments may comprise software. In such an embodiment, the program instructions that implement the methods and/or mechanisms may be conveyed or stored on a non-transitory computer readable medium. Numerous types of media which are configured to store program instructions are available and include hard disks, floppy disks, CD-ROM, DVD, flash memory, Programmable ROMs (PROM), random access memory (RAM), and various other forms of volatile or non-volatile storage.

[0060] In various embodiments, one or more portions of the methods and mechanisms described herein may form part of a cloud-computing environment. In such embodiments, resources may be provided over the Internet as services according to one or more various models. Such models may include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In IaaS, computer infrastructure is delivered as a service. In such a case, the computing equipment is generally owned and operated by the service provider. In the PaaS model, software tools and underlying equipment used by developers to develop software solutions may be provided as a service and hosted by the service provider. SaaS typically includes a service provider licensing software as a service on demand. The service provider may host the software, or may deploy the software to a customer for a given period of time. Numerous combinations of the above models are possible and are contemplated.

[0061] Although the embodiments above have been described in considerable detail, numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

1. A storage controller configured to:

generate an address for an I/O operation that is to be directed to one or more storage devices in a storage system; and

issue an I/O operation to the one or more storage devices using the generated address, wherein the generated address exceeds the range of actual addressable locations within the storage device.

2. The storage controller of claim **1** wherein the I/O operation is issued using a protocol that addresses blocks with a fixed-width address.

3. The storage controller of claim **2**, wherein the first protocol is Serial Advanced Technology Attachment (SATA), Small Computer System Interface (SCSI) or Serial Attached SCSI (SAS).

4. The storage controller of claim **1** wherein the storage controller includes a translation layer that maintains a mapping table to map addresses in an exported address space to addressable storage locations on the one or more storage devices, wherein the one or more storage devices include a first number of addressable storage locations and the exported address space has a second number of addressable storage locations, wherein the second number is greater than the first number.

5. The storage controller of claim **4** wherein the mapping table is updated during garbage collection.

6. The storage controller of claim **4** wherein the mapping table maps object addresses to physical addresses in the one or more storage devices.

7. The storage controller of claim **1** wherein the storage controller is coupled to multiple storage devices, and the storage controller is further configured to distribute I/O operations amongst the multiple storage devices.

8. A method comprising:

generating an address for an I/O operation that is to be directed to one or more storage devices in a storage system; and

issuing an I/O operation to the one or more storage devices using the generated address, wherein the generated address exceeds the range of actual addressable locations within the storage device.

9. The method as recited in claim **8**, wherein the I/O operation is issued using a protocol that addresses blocks with a fixed-width address.

10. The method as recited in claim **9**, wherein the first protocol is Serial Advanced Technology Attachment (SATA), Small Computer System Interface (SCSI) or Serial Attached SCSI (SAS).

11. The method as recited in claim **8**, further comprising maintaining a mapping table to map addresses in an exported address space to addressable storage locations on the one or more storage devices, wherein the one or more storage devices include a first number of addressable storage locations and the exported address space has a second number of addressable storage locations, wherein the second number is greater than the first number.

12. The method as recited in claim **11**, wherein the mapping table is updated during garbage collection.

13. The method as recited in claim **11**, wherein the mapping table maps object addresses to physical addresses in the one or more storage devices.

14. The method as recited in claim **8**, further comprising distributing I/O operations amongst the multiple storage devices.

15. A non-transitory computer readable storage medium comprising program instructions, wherein the program instructions are executable to:

generate an address for an I/O operation that is to be directed to one or more storage devices in a storage system; and

issue an I/O operation to the one or more storage devices using the generated address, wherein the generated address exceeds the range of actual addressable locations within the storage device.

16. The non-transitory computer readable storage medium as recited in claim **15**, wherein the I/O operation is issued using a protocol that addresses blocks with a fixed-width address.

17. The non-transitory computer readable storage medium as recited in claim **16**, wherein the first protocol is Serial Advanced Technology Attachment (SATA), Small Computer System Interface (SCSI) or Serial Attached SCSI (SAS).

18. The non-transitory computer readable storage medium as recited in claim **15**, wherein the program instructions are further executable to maintain a mapping table to map addresses in an exported address space to addressable storage locations on the one or more storage devices, wherein the one or more storage devices include a first number of addressable storage locations and the exported address space has a second number of addressable storage locations, wherein the second number is greater than the first number.

19. The non-transitory computer readable storage medium as recited in claim **18**, wherein the mapping table is updated during garbage collection.

20. The non-transitory computer readable storage medium as recited in claim **18**, wherein the mapping table maps object addresses to physical addresses in the one or more storage devices.

* * * * *