United States Patent Application Publication 20250251860
Kind Code A1
Publication Date August 07, 2025
Inventor(s) SINGH; Rohit Shankar et al.

METHODS FOR HANDLING INPUT-OUTPUT OPERATIONS IN ZONED STORAGE SYSTEMS AND DEVICES THEREOF

Abstract

The disclosed technology relates to managing input-output operation in a zoned storage system includes identifying a first physical zone and a second physical zone within a zoned namespace solid-state drive associated with a logical zone to perform a received write operation. Data to be written in the received write operation is temporarily staged in a zone random write area associated with the identified second physical zone. Based a storage threshold of the zone random write area, a determination is made regarding when to transfer temporarily staged data to be written area to the identified second physical zone. When the storage threshold of the zone random write area determined to have exceeded, temporarily staged data to be written is transferred to the identified second physical zone.

Inventors: **SINGH; Rohit Shankar (Cary, NC), Doucette; Douglas P. (San Diego, CA), Gole; Abhijeet Prakash (Cupertino, CA), Deshpande; Prathamesh (Folsom, CA)**

Applicant: **NetApp, Inc.** (San Jose, CA)

Family ID: **76197551**

Appl. No.: **19/190994**

Filed: **April 28, 2025**

Related U.S. Application Data

parent US continuation 18343149 20230628 PENDING child US 19190994
parent US continuation 16858019 20200424 parent-grant-document US 11789611 child US 18343149

Publication Classification

Int. Cl.: **G06F3/06** (20060101)

U.S. Cl.:

CPC **G06F3/0608** (20130101); **G06F3/0631** (20130101); **G06F3/0679** (20130101);

Background/Summary

[0001] This application is a continuation of U.S. patent application Ser. No. 18/343,149, filed Jun. 28, 2023, which is a continuation of U.S. patent application Ser. No. 16/858,019, filed Apr. 24, 2020 (now U.S. Pat. No. 11,789,611, issued Oct. 17, 2023), each of which is incorporated herein by reference in its entirety.

FIELD

[0002] The present technology pertains to the field of storage management, and particularly, directed to a method for handling input-output operations in zoned storage systems and devices.

BACKGROUND

[0003] A solid-state drive (SSD) is a data storage device that uses non-volatile solid-state memory to store persistent digitally encoded data. A solid-state drive can be configured to emulate a hard disk drive, i.e., a device that stores persistent digitally encoded data on the magnetic surfaces of rapidly rotating platters, and can be used to replace a hard disk drive in many applications. However, one difference between a solid-state drive and a hard disk drive is that solid-state drives do not have a spinning magnetic platter or actuator arm as used in hard disk drives. Therefore, solid-state drives are more rugged than hard disk drives and do not have the same operational delays.

[0004] Unfortunately, one disadvantage associated with a solid-state drive when compared to a hard disk drive is that once an erase block has been programmed with some data, the same erase block cannot be written to again until the erase block has been sequentially erased. To overcome this limitation, modern solid-state drives run complex pieces of software known as flash-translation layers to hide these complications by presenting a uniformly-mutable block interface similar to a hard-drive disk. Unfortunately, this layer requires a significant amount of space to manage its block indirection metadata and garbage collection activities.

Description

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. **1** is a block diagram of a network environment with exemplary data storage apparatuses each including a node computing device;

[0006] FIG. **2** is a block diagram of an exemplary one of the node computing devices shown in FIG. **1**;

[0007] FIG. **3** is a flowchart of an exemplary method for handling write request in a zoned storage system;

[0008] FIG. **4** is an exemplary block diagram illustrating the relation between a logical zone and a physical zone using a mapping table;

[0009] FIG. **5** is an exemplary block diagram illustrating a previous physical zone and a physical zone;

[0010] FIGS. **6**A-**6**F are an exemplary block diagrams illustrating a write operation using a zone random write area;

[0011] FIG. **7** is an exemplary block diagram illustrating erasing of data in the previous physical zone;

[0012] FIGS. **8**A-**8**C is an exemplary block diagram illustrating an implicit commit operation; and

[0013] FIG. **9** is flowchart of an exemplary method for handling a read operation while performing a write operation in the zone storage system.

DETAILED DESCRIPTION

[0014] A clustered network environment **100** that may implement one or more aspects of the technology described and illustrated herein is shown in FIG. **1**. The clustered network environment **100** includes data storage apparatuses **102**(**1**)-**102**(*n*) that are coupled over a cluster fabric **104** facilitating communication between the data storage apparatuses **102**(**1**)-**102**(*n*) (and one or more modules, components, etc. therein, such as, node computing devices **106**(**1**)-**106**(*n*), for example), although any number of other elements or components can also be included in the clustered network environment **100** in other examples.

[0015] This technology provides a number of advantages including methods, non-transitory computer readable media, and devices that more effectively and efficiently handle input-output operations in zoned storage systems. With the disclosed technology, a pair-tuple of physical zones for each logical zone is used to manage the input-output operations. In the disclosed technology, one physical zone represents the logical older copy of the data, whereas the other physical zone represents the logical newer copy of the data. Once the zone has been fully written, the disclosed technology detaches the older physical zone from the logical zone and erasure of the data present in the older physical zone is scheduled so that the older physical zone can be used to perform other input-output operations. Additionally, the disclosed technology uses a zone random write area (or a buffer space) that can be used as a staging buffer to manage random input-output operations so that random input-output operations do not trigger the rewriting or erasing of the data in the access unit of the solid-state drive. By using the techniques illustrated below, the disclosed technology is able to support random input-output operations on the solid-state drives without significant operational delay.

[0016] In this example, node computing devices **106**(**1**)-**106**(*n*) can be primary or local storage controllers or secondary or remote storage controllers that provide client devices **108**(**1**)-**108**(*n*), with access to data stored within data storage devices **110**(**1**)-**110**(*n*). The data storage apparatuses **102**(**1**)-**102**(*n*) and/or node computing device **106**(**1**)-**106**(*n*) of the examples described and illustrated herein are not limited to any particular geographic areas and can be clustered locally and/or remotely. Thus, in one example the data storage apparatuses **102**(**1**)-**102**(*n*) and/or node computing device **106**(**1**)-**106**(*n*) can be distributed over a plurality of storage systems located in a plurality of geographic locations. In another example, a clustered network can include data storage apparatuses **102**(**1**)-**102**(*n*) and/or node computing device **106**(**1**)-**106**(*n*) residing in a same geographic location (e.g., in a single onsite rack).

[0017] In the illustrated example, one or more of the client devices **108**(**1**)-**108**(*n*), which may be, for example, personal computers (PCs), computing devices or storage (e.g., storage servers), and other computers or peripheral devices, are coupled to the respective data storage apparatuses **102**(**1**)-**102**(*n*) by storage network connections **112**(**1**)-**112**(*n*). Network connections **112**(**1**)-**112**(*n*) may include a local area network (LAN) or wide area network (WAN), for example, that utilizes Network Attached Storage (NAS) protocols, such as a Common Internet File System (CIFS) protocol or a Network File System (NFS) protocol to exchange data packets, a Storage Area Network (SAN) protocol, such as Small Computer System Interface (SCSI) or Fiber Channel Protocol (FCP), an object protocol, such as S3, etc.

[0018] Illustratively, the client devices **108**(**1**)-**108**(*n*) may be general-purpose computers running applications, and may interact with the data storage apparatuses **102**(**1**)-**102**(*n*) using a client/server model for exchange of information. That is, the client devices **108**(**1**)-**108**(*n*) may request data from the data storage apparatuses **102**(**1**)-**102**(*n*) (e.g., data on one of the data storage devices **110**(**1**)-**110**(*n*) managed by a network storage controller configured to process I/O commands issued by the client devices **108**(**1**)-**108**(*n*)), and the data storage apparatuses **102**(**1**)-**102**(*n*) may return results of the request to the client devices **108**(**1**)-**108**(*n*) via the storage network connections

112(**1**)-**112**(*n*).

[0019] The node computing devices **106**(**1**)-**106**(*n*) of the data storage apparatuses **102**(**1**)-**102**(*n*) can include network or host nodes that are interconnected as a cluster to provide data storage and management services, such as to an enterprise having remote locations, cloud storage (e.g., a storage endpoint may be stored within a data cloud), etc., for example. Such a node computing device **106**(**1**)-**106**(*n*) can be a device attached to the fabric **104** as a connection point, redistribution point, or communication endpoint, for example. One or more of the node computing devices **106**(**1**)-**106**(*n*) may be capable of sending, receiving, and/or forwarding information over a network communications channel, and could comprise any type of device that meets any or all of these criteria.

[0020] In an example, the node computing device **106**(**1**) may be located on a first storage site and the node computing device **106**(*n*) may be located at a second storage site. The node computing devices **106**(**1**) and **106**(*n*) may be configured according to a disaster recovery configuration whereby a surviving node provides switchover access to the storage devices **110**(**1**)-**110**(*n*) in the event a disaster occurs at a disaster storage site (e.g., the node computing device **106**(**1**) provides client device **108**(*n*) with switchover data access to storage devices **110**(*n*) in the event a disaster occurs at the second storage site). In other examples, the node computing device **106**(*n*) can be configured according to an archival configuration and/or the node computing devices **106**(**1**)-**106**(*n*) can be configured based on another type of replication arrangement (e.g., to facilitate load sharing). Additionally, while two node computing devices **106** are illustrated in FIG. **1**, any number of node computing devices or data storage apparatuses can be included in other examples in other types of configurations or arrangements.

[0021] As illustrated in the clustered network environment **100**, node computing devices **106**(**1**)-**106**(*n*) can include various functional components that coordinate to provide a distributed storage architecture. For example, the node computing devices **106**(**1**)-**106**(*n*) can include network modules **114**(**1**)-**114**(*n*) and disk modules **116**(**1**)-**116**(*n*). Network modules **114**(**1**)-**114**(*n*) can be configured to allow the node computing devices **106**(**1**)-**106**(*n*) (e.g., network storage controllers) to connect with client devices **108**(**1**)-**108**(*n*) over the storage network connections **112**(**1**)-**112**(*n*), for example, allowing the client devices **108**(**1**)-**108**(*n*) to send input-output operations to the node computing devices **106**(**1**)-**106**(*n*).

[0022] Further, the network modules **114**(**1**)-**114**(*n*) can provide connections with one or more other components through the cluster fabric **104**. For example, the network module **114**(**1**) of node computing device **106**(**1**) can access the data storage device **110**(*n*) by sending a request via the cluster fabric **104** through the disk module **116**(*n*) of node computing device **106**(*n*) when the node computing device **106**(*n*) is available. Alternatively, when the node computing device **106**(*n*) fails, the network module **114**(**1**) of node computing device **106**(**1**) can access the data storage device **110**(*n*) directly via the cluster fabric **104**. The cluster fabric **104** can include one or more local and/or wide area computing networks embodied as Infiniband, Fibre Channel (FC), or Ethernet networks, for example, although other types of networks supporting other protocols can also be used.

[0023] Disk modules **116**(**1**)-**116**(*n*) can be configured to connect data storage devices **110**(**1**)-**110**(*n*), such as disks or arrays of disks, SSDs, flash memory, or some other form of data storage, to the node computing devices **106**(**1**)-**106**(*n*). Often, disk modules **116**(**1**)-**116**(*n*) communicate with the data storage devices **110**(**1**)-**110**(*n*) according to the SAN protocol, such as SCSI, FCP, SAS, NVMe, NVMe-oF for example, although other protocols can also be used. Thus, as seen from an operating system on either of node computing devices **106**(**1**)-**106**(*n*), the data storage devices **110**(**1**)-**110**(*n*) can appear as locally attached. In this manner, different node computing devices **106**(**1**)-**106**(*n*), etc. may access data blocks through the operating system, rather than expressly requesting abstract files.

[0024] While the clustered network environment **100** illustrates an equal number of network modules **114**(**1**)-**114**(*n*) and disk modules **116**(**1**)-**116**(*n*), other examples may include a differing number of these modules. For example, there may be a plurality of network and disk modules interconnected in a cluster that do not have a one-to-one correspondence between the network and disk modules. That is, different node computing devices can have a different number of network and disk modules, and the same node computing device can have a different number of network modules than disk modules.

[0025] Further, one or more of the client devices **108**(**1**)-**108**(*n*) can be networked with the node computing devices **106**(**1**)-**106**(*n*) in the cluster, over the storage connections **112**(**1**)-**112**(*n*). As an example, respective client devices **108**(**1**)-**108**(*n*) that are networked to a cluster may request services (e.g., exchanging of information in the form of data packets) of node computing devices **106**(**1**)-**106**(*n*) in the cluster, and the node computing devices **106**(**1**)-**106**(*n*) can return results of the requested services to the client devices **108**(**1**)-**108**(*n*). In one example, the client devices **108**(**1**)-**108**(*n*) can exchange information with the network modules **114**(**1**)-**114**(*n*) residing in the node computing devices **106**(**1**)-**106**(*n*) (e.g., network hosts) in the data storage apparatuses **102**(**1**)-**102**(*n*).

[0026] In one example, the storage apparatuses **102**(**1**)-**102**(*n*) host aggregates corresponding to physical local and remote data storage devices, such as local flash or disk storage in the data storage devices **110**(**1**)-**110**(*n*), for example. One or more of the data storage devices **110**(**1**)-**110**(*n*) can include mass storage devices, such as disks of a disk array. The disks may comprise any type of mass storage devices, including but not limited to magnetic disk drives, flash memory, SSDs, storage class memories and any other similar media adapted to store information, including, for example, data (D) and/or parity (P) information. In this example, the SSDs in the data storage devices **110**(**1**)-**110**(*n*) are arranged in a zoned namespace configuration (where the logical address space of the namespace is divided into zones) where a zone is a portion of the namespace (contiguous LBA range) with specific write access rules.

[0027] The aggregates include volumes **118**(**1**)-**118**(*n*) in this example, although any number of volumes can be included in the aggregates. The volumes **118**(**1**)-**118**(*n*) are virtual data stores that define an arrangement of storage and one or more file systems within the clustered network environment **100**. Volumes **118**(**1**)-**118**(*n*) can span a portion of a disk or other storage device, a collection of disks, or portions of disks, for example, and typically define an overall logical arrangement of file storage. In one example volumes **118**(**1**)-**118**(*n*) can include stored data as one or more files or objects that reside in a hierarchical directory structure within the volumes **118**(**1**)-**118**(*n*). Volumes **118**(**1**)-**118**(*n*) are typically configured in formats that may be associated with particular storage systems, and respective volume formats typically comprise features that provide functionality to the volumes **118**(**1**)-**118**(*n*), such as providing an ability for volumes **118**(**1**)-**118**(*n*) to form clusters.

[0028] In one example, to facilitate access to data stored on the disks or other structures of the data storage device **110**(**1**)-**110**(*n*), a file system (e.g., write anywhere file system (WAFL)) may be implemented that logically organizes the information as a hierarchical structure of directories and files. In this example, respective files may be implemented as a set of disk blocks configured to store information, whereas directories may be implemented as specially formatted files in which information about other files and directories are stored.

[0029] Data can be stored as files or objects within a physical volume and/or a virtual volume, which can be associated with respective volume identifiers, such as file system identifiers (FSIDs). The physical volumes correspond to at least a portion of physical storage devices, such as the data storage device **110**(**1**)-**110**(*n*) (e.g., a Redundant Array of Independent (or Inexpensive) Disks (RAID system)) whose address, addressable space, location, etc. does not change. Typically the location of the physical volumes does not change in that the (range of) address(es) used to access it generally remains constant.

[0030] Virtual volumes, in contrast, are stored over an aggregate of disparate portions of different

physical storage devices. Virtual volumes may be a collection of different available portions of different physical storage device locations, such as some available space from disks, for example. It will be appreciated that since the virtual volumes are not "tied" to any one particular storage device, virtual volumes can be said to include a layer of abstraction or virtualization, which allows them to be resized and/or flexible in some regards.
[0031] Further, virtual volumes can include one or more logical unit numbers (LUNs), directories, Qtrees, and/or files. Among other things, these features, but more particularly the LUNS, allow the disparate memory locations within which data is stored to be identified, for example, and grouped as a data storage unit. As such, the LUNs may be characterized as constituting a virtual disk or drive upon which data within the virtual volumes is stored within an aggregate. For example, LUNs are often referred to as virtual disks, such that they emulate a hard drive, while they actually comprise data blocks stored in various parts of a volume.
[0032] In one example, the data storage devices **110**(**1**)-**110**(*n*) can have one or more physical ports, wherein each physical port can be assigned a target address (e.g., SCSI target address). To represent respective volumes, a target address on the data storage devices **110**(**1**)-**110**(*n*) can be used to identify one or more of the LUNs. Thus, for example, when one of the node computing devices **106**(**1**)-**106**(*n*) connects to a volume, a connection between the one of the node computing devices **106**(**1**)-**106**(*n*) and one or more of the LUNs underlying the volume is created.
[0033] In one example, respective target addresses can identify multiple of the LUNs, such that a target address can represent multiple volumes. The I/O interface, which can be implemented as circuitry and/or software in a storage adapter or as executable code residing in memory and executed by a processor, for example, can connect to volumes by using one or more addresses that identify the one or more of the LUNs.
[0034] Referring to FIG. **2**, node computing device **106**(**1**) in this particular example includes processor(s) **200**, a memory **202**, a network adapter **204**, a cluster access adapter **206**, and a storage adapter **208** interconnected by a system bus **210**. The node computing device **106** also includes a storage operating system **212** installed in the memory **206** that can, for example, implement a Redundant Array of Independent (or Inexpensive) Disks (RAID) data loss protection and recovery scheme to optimize a reconstruction process of data of a failed disk or drive in an array. In some examples, the node computing device **106**(*n*) is substantially the same in structure and/or operation as node computing device **106**(**1**), although the node computing device **106**(*n*) can include a different structure and/or operation in one or more aspects than the node computing device **106**(**1**) in other examples.
[0035] The storage operating system **212** can also manage communications for the node computing device **106**(**1**) among other devices that may be in a clustered network, such as attached to a cluster fabric **104**. Thus, the node computing device **106**(**1**) can respond to client device requests to manage data on one of the data storage devices **110**(**1**)-**110**(*n*) (e.g., or additional clustered devices) in accordance with the client device requests.
[0036] The storage operating system **212** can also establish one or more file systems including software code and data structures that implement a persistent hierarchical namespace of files and directories, for example. As an example, when a new data storage device (not shown) is added to a clustered network system, the storage operating system **212** is informed where, in an existing directory tree, new files associated with the new data storage device are to be stored. This is often referred to as "mounting" a file system.
[0037] In the example node computing device **106**(**1**), memory **202** can include storage locations that are addressable by the processor(s) **200** and adapters **204**, **206**, and **208** for storing related software application code and data structures. The processor(s) **200** and adapters **204**, **206**, and **208** may, for example, include processing elements and/or logic circuitry configured to execute the software code and manipulate the data structures.
[0038] The storage operating system **212**, portions of which are typically resident in the memory

202 and executed by the processor(s) **200**, invokes storage operations in support of a file service implemented by the node computing device **106**(**1**). Other processing and memory mechanisms, including various computer readable media, may be used for storing and/or executing application instructions pertaining to the techniques described and illustrated herein. For example, the storage operating system **212** can also utilize one or more control files (not shown) to aid in the provisioning of virtual machines.

[0039] Accordingly, the examples may be embodied as one or more non-transitory computer readable media having machine or processor-executable instructions stored thereon for one or more aspects of the present technology, as described and illustrated by way of the examples herein, which when executed by the processor(s) **200**, cause the processor(s) **200** to carry out the steps necessary to implement the methods of this technology, as described and illustrated with the examples herein. In some examples, the executable instructions are configured to perform one or more steps of a method, such as one or more of the exemplary methods described and illustrated later with reference to FIGS. **3**-**9**, for example.

[0040] The network adapter **204** in this example includes the mechanical, electrical and signaling circuitry needed to connect the node computing device **106**(**1**) to one or more of the client devices **108**(**1**)-**108**(*n*) over storage network connections **112**(**1**)-**112**(*n*), which may comprise, among other things, a point-to-point connection or a shared medium, such as a local area network. In some examples, the network adapter **204** further communicates (e.g., using TCP/IP) via the fabric **104** and/or another network (e.g. a WAN) (not shown) with cloud storage devices to process storage operations associated with data stored thereon.

[0041] The storage adapter **208** cooperates with the storage operating system **212** executing on the node computing device **106**(**1**) to access information requested by one of the client devices **108**(**1**)-**108**(*n*) (e.g., to access data on a data storage device **110**(**1**)-**110**(*n*) managed by a network storage controller). The information may be stored on any type of attached array of writeable media such as magnetic disk drives, SSDs, and/or any other similar media adapted to store information.

[0042] In the exemplary data storage devices **110**(**1**)-**110**(*n*), information can be stored in data blocks on disks. The storage adapter **208** can include input/output (I/O) interface circuitry that couples to the disks over an I/O interconnect arrangement, such as a storage area network (SAN) protocol (e.g., Small Computer System Interface (SCSI), iSCSI, hyperSCSI, Fiber Channel Protocol (FCP)). The information is retrieved by the storage adapter **208** and, if necessary, processed by the processor(s) **200** (or the storage adapter **208** itself) prior to being forwarded over the system bus **210** to the network adapter **204** (and/or the cluster access adapter **206** if sending to another node computing device in the cluster) where the information is formatted into a data packet and returned to a requesting one of the client devices **108**(**1**)-**108**(*n*), or alternatively sent to another node computing device attached via the cluster fabric **104**. In some examples, a storage driver **214** in the memory **202** interfaces with the storage adapter to facilitate interactions with the data storage devices **110**(**1**)-**110**(*n*), as described and illustrated in more detail later with reference to FIGS. **3**-**9**.

[0043] Now, an exemplary method for handling input-output operations in zoned storage systems will be illustrated with reference to FIGS. **3**-**9**. Particularly with reference to FIG. **3**, an exemplary method for managing a write request begins at step **305** where the node computing device **106**(**1**) receives a data write request from a write anywhere file system (WAFL), although the node computing device **106**(**1**) can receive the write request from one of the client devices **108**(**1**). While this example illustrates the node computing device **106**(**1**) of the node computing devices **106**(**1**)-**106**(*n*) performing the steps illustrated in FIGS. **3**-**9**, it is to be understood that other node computing devices of the plurality of node computing devices **106**(**1**)-**106**(*n*) can perform the steps illustrated in FIGS. **3**-**9**.

[0044] In step **310**, the node computing device **106**(**1**) identifies a logical zone to write the data within the zoned namespace SSDs within the data storage devices **110**(**1**)-**110**(*n*), based on the data present in the write request, although the node computing device **106**(**1**) can identify the logical zone based on the logical block address provided in the received write request. In this example, a logical zone is a portion of the SSD namespace (logical grouping of volumes) with specific write access rules and maps to one or more erase blocks on the SSDs. Additionally in this example, the logical zone is associated with a physical zone where the physical zone is a physical location within the zoned namespace SSDs where data can be read, written, or erased. The mapping of the logical zone and the physical zone is present within the mapping table stored within the memory **202**, although the mapping table may be present at other memory locations such as the zoned namespace SSDs. An example step **310** is illustrated in FIG. **4** where the logical zone **405** includes logical zone 0, logical zone 1, and logical zone 2 and a mapping table **410** that includes a mapping of the logical zone to the physical zone that points to the physical location within the zoned namespace SSDs. By way of example, logical zone 0 within the mapping table **410** is associated with the physical zone **100**, logical zone 1 is associated with the physical location **1600**, and the logical zone 2 is associated with the physical zone **102**.

[0045] In step **315**, the node computing device **106**(**1**) determines if the identified logical zone has a corresponding previous physical zone using the mapping table. As illustrated above, each logical zone generally has a corresponding physical zone assigned and the correlation between the logical zone and the previous physical zone is present within the mapping table. However, when there is no previous physical zone associated with the logical zone, the mapping table would not include the data associated with the previous physical zone. In other words, the mapping table would include a blank for the previous physical zone when there is no previous physical zone associated with the logical zone. Accordingly, if the node computing device **106**(**1**) determines that the identified logical zone does not have a previous physical zone, then the No branch is taken to step **320**.

[0046] In step **320**, the node computing device **106**(**1**) identifies a new empty physical zone from a list of unused physical zones. Additionally, the node computing device **106**(**1**) moves the physical zone that is associated with the logical zone from the mapping table to the previous physical zone column and assigns the identified new physical zone as the physical zone the exemplary flow proceeds to step **325**. An example of the step **320** is illustrated in FIG. **5** where there is a previous physical zone **1600** for the identified logical zone and a physical zone **500** is identified from the list of unused physical zones. Additionally, the mapping table **520** in FIG. **5** illustrates the mapping between the logical zones, the previous physical zone, and the physical zone. By way of example, the logical zone 1 correlates to previous physical zone **1600** and the physical zone **500** in the mapping table **520**.

[0047] Referring back to FIG. **3**, back in step **315**, if the node computing device **106**(**1**) determines that there is a previous physical zone corresponding to the identified logical zone, then the Yes branch is taken to step **325**.

[0048] In step **325**, the node computing device **106**(**1**) performs the received write operation and the step of performing the received write operation will now be further illustrated with reference to FIG. **6**A. First, the node computing device **106**(**1**) temporarily stages the received write operation into a zone random write area that is associated with the physical zone. In this example, the zone random write area assists with temporarily staging the received write data. The size of the zone random write area is substantially smaller when compared to the size of the physical zone as the zone random write area is only there to temporarily stage the received data. Furthermore, the zone random write area is present within the ZNS SSDs, although the zone random write area can be present at other memory locations. Furthermore, the zone random write area might be backed by persistent memory or it might be backed by high-performance NAND flash, by way of example. An example of this step of temporarily staging the received write operation in the zone random write area is illustrated in FIG. **6**A. By way of example, if the node computing device **106**(**1**) receives a write operation to write data B**1**′, the node computing device **106**(**1**) temporarily stages the received data in the zone random write area **620** that is associated with the physical zone **500**.

[0049] Referring to FIG. **3**, in step **330**, the node computing device **106**(**1**) determines if there are any missing blocks of data in addition to the data that is present in the zone random write area. In this example, a logical zone can include multiple blocks and a block in this example is a smallest memory unit within the logical zone. Accordingly, if the node computing device **106**(**1**) determines that there are missing blocks of data that is to be written, then the Yes branch is taken to step **335**.
[0050] In step **335**, the node computing device **106**(**1**) reads the missing blocks from the previous physical zone associated with the physical zone. An example of this step will now be further illustrated with reference to the mapping table **520** in FIG. **5**. In this example, the logical zone 1 is mapping to the previous physical zone **1600** and the physical zone **500**. Accordingly, when the node computing device **106**(**1**) determines that there are missing blocks, the node computing device **106**(**1**) will read the missing blocks from the previous physical zone **1600** and the exemplary flow proceeds back to step **325**.
[0051] Referring to FIG. **3**, in step **330**, if the node computing device **106**(**1**) determines that there are no missing blocks, then the No branch is taken to step **340**. In step **340**, the node computing device **106**(**1**) determines if a commit boundary has been reached. In this example, a commit boundary is the threshold capacity of data that can be stored within the zone random write area and is based on the minimum commit size of the underlying device. In this example, the node computing device **106**(**1**) periodically transfers the data from the zone random write area to the corresponding physical zone once the zone random write area has been filled, although the commit boundary is typically smaller than the zone random write area. Accordingly, when the node computing device **106**(**1**) determines that the commit boundary has been reached, then the Yes branch is taken to step **345**.
[0052] In step **345**, the node computing device **106**(**1**) performs the commit operation. In this example, a commit operation relates to a command that carries instructions to transfer the data from the zone random write area to the physical zone. An example of step **345** will be further illustrated with reference to FIG. **6**B where the node computing device **106**(**1**) commits the received write data B**1**′ along with the missing data blocks read from the previous physical zone **1600** into the physical zone **500**. Additionally in this example, the node computing device advances the zone random write area to point to the next location within the physical zone **500** where the data can be written.
[0053] Referring to FIG. **3**, in step **350**, the node computing device **106**(**1**) determines if the end of the physical zone has been reached after performing the commit operation. As illustrated above, the physical zone includes blocks to which data can be written. When only a portion of the blocks are consumed after commit the data from the zone random write area, there will be remaining blocks that are unwritten and therefore would not have reached the end of the physical zone. An example of this step **350** will now be illustrated with reference to FIG. **6**B. As illustrated in FIG. **6**B, the node computing device **106**(**1**) commits the data B**1**′ along with the missing data from the zone random write area **620** and only a portion of the blocks within the physical zone **500** are consumed. Accordingly, a portion of the physical zone **500** includes empty blocks and therefore has performing the commit operation has not resulted in reaching the end of the physical zone **500**. Referring to FIG. **3**, if the node computing device **106**(**1**) determines that it is not the end of the zone, then the No branch is taken to step **305** where subsequent write requests are received. Upon receiving subsequent write requests, steps **310**-**345** are again performed and an illustrative example will now be described with reference to FIGS. **6**C-**6**E.
[0054] In FIG. **6**E, the node computing device **106**(**1**) receives a subsequent write data B**2**′ that is temporarily staged within the zone random write area **620**. Additionally, the node computing device **106**(**1**) determines if there are any missing blocks and if there are any missing blocks, then the node computing device **106**(**1**) reads the previous physical zone **1600** for the missing blocks. Eventually, the node computing device **106**(**1**) upon reaching the commit boundary performs the commit operation by transferring the received written data B**2**′ along with the missing data to the physical

zone **500** as illustrated in FIG. **6**D. Subsequently, the node computing device **106**(**1**) advances the zone random write area **620** to point to the next location where the data can be written as illustrated in FIG. **6**D. Further, the node computing device **106** then determines if the end of the zone has been reached and as illustrated in FIG. **6**D, there are still empty blocks and therefore the end of the physical zone **500** has not been reached.
[0055] Subsequently, the node computing device **106**(**1**) receives another write request to write data B**3**′. As illustrated above, the received write data B**3**′ is temporarily staged within the zone random write area **620** as illustrated in FIG. **6**E. Additionally, the node computing device **106**(**1**) determines if there are any missing blocks and if there are any missing blocks, then the node computing device **106**(**1**) reads the previous physical zone **1600** for the missing blocks. Eventually, the node computing device **106**(**1**) upon reaching the commit boundary performs the commit operation by transferring the received written data B**3**′ along with the missing data to the physical zone **500** as illustrated in FIG. **6**F. Again, the node computing device **106**(**1**) determines performing the commit operation has resulted in reaching the end of the zone. As illustrated in FIG. **6**F, the node computing device **106**(**1**) after performing the commit operation of B**3**′ along with the missing data has resulted in reaching the end of the physical zone **500**.
[0056] Referring to FIG. **3**, if in step **350**, the node computing device **106**(**1**) determines that the end of the zone is reached, then the Yes branch is taken to step **355**. In step **355**, the node computing device **106**(**1**) erases the data in previous physical zone associated with the identified logical zone. Additionally, the node computing device **106**(**1**) updates the mapping table to delete the reference to the previous physical zone. An example of step **350** will be illustrated with reference to FIG. **7**. As illustrated in FIG. **7**, the node computing device **106**(**1**) erases the data present in the previous physical zone and deletes the entry in to mapping table **520**. Once erased, physical zone **1600** will be available for subsequent reuse. The exemplary flow proceeds back to step **305** where subsequent write requests are received.
[0057] While step **345** and FIGS. **6**B, **6**D, and **6**F illustrates one example of an explicit commit operation, an alternative commit operation, i.e., implicit commit, will now be illustrated with reference to FIG. **8**A-**8**B. An example in FIG. **8**A illustrates the zone random write area including data M and N to be written to the physical zone in the ZNS SSD on a solid-state drive that supports implicit commit. In FIG. **8**A, the zone random write area is 4 blocks and the minimum commit size is 2 blocks. By temporarily staging the data M and N within the zone random write area, the zone random write area has reached its window size (or the threshold capacity). Accordingly, when the zone random write area receives subsequent data O to be written beyond the boundary of the zone random write area as illustrated in FIG. **8**B, the subsequent data is temporarily stalled before writing to the zone random write area. Next, the node computing device **106**(**1**) implicitly commits or transfers data M to the physical zone, moves the write pointer by the minimum commit size, and then temporarily stages the stalled write data O into the zone random write area as illustrated in FIG. **8**C.
[0058] An exemplary method for managing a read request in zoned storage systems will be illustrated with reference to the exemplary flowchart in FIG. **9**. In step **905**, the node computing device **106**(**1**) receives a read request from the write anywhere file system, although the node computing device **106**(**1**) can receive the read request one of the client devices **108**(**1**) while performing a previously received write operation using the technique illustrated above. In this example, the received read request includes the logical zone where the data is present, although the data read request can include other types and/or amounts of information such as the logical block address from where the data is to be read.
[0059] In step **910**, the node computing device **106**(**1**) identifies the logical zone and the corresponding previous physical zone along with the physical zone using the mapping table stored within the memory **206**, although the node computing device **106**(**1**) can use other techniques to identify the physical zone. In this example, the mapping table is used to map the logical zone to the

corresponding previous physical zone and the physical zone.
[0060] In step **915**, the node computing device **106**(**1**) determines if the received read request is before the current position of a write pointer of the physical zone where the write pointer indicates the current position within the physical zone where the data is to be committed when destaged from the zone random write area. Accordingly, when the node computing device **106**(**1**) determines that the received read operation is before the write pointer, then the Yes branch is taken to step **920**.
[0061] In step **920**, the node computing device **106**(**1**) reads the requested data from the physical zone, provides it to the requesting client device **108**(**1**) and the exemplary method ends at step **940**.
[0062] However, back in step **915**, if the node computing device **106**(**1**) determines that the received read request is not before the write pointer, then the No branch is taken to step **925**. In step **925**, the node computing device **106**(**1**) determines if the received read request is after the end of the zone random write area. As illustrated above in FIGS. **6**A-**6**F, the start of the zone random write area is at a location where the data can be next committed to in the physical zone. In other words, the position before the zone random write area is where the data has been transferred to the physical zone and the position after the start of the zone random write area is where the data can be written. Accordingly, when the node computing device **106**(**1**) determines that the read is after the end of the zone random write area, then the Yes branch is taken to step **930**.
[0063] In step **930**, the node computing device **106**(**1**) performs the read operation from the previous physical zone and the exemplary flow proceeds to step **940** where the exemplary method ends.
[0064] However, back in step **925**, if the node computing device **106**(**1**) determines that the read is not after the end of the zone random write area, then the No branch is taken to step **935**. In step **935**, the node computing device **106**(**1**) refers to zone random write area bitmap which includes the data associated with whether the data block within the zone has been written to the zone random write area. Accordingly, if the node computing device **106**(**1**) determines that the data block has been written, then the exemplary flow proceeds to step **920** where the read operation is performed from the block that is written in the physical zone and will implicitly retrieve the data from the zone random write area attached to the physical zone. However, when the node computing device from the zone random write area determines that the block has not been written, then the exemplary flow proceeds to step **930** where the read operation is performed by the previous physical zone and the exemplary method ends at step **940**.
[0065] Accordingly, as illustrated and described by way of the examples here, the above illustrated technology uses a pair-tuple of physical zones for each logical zone is used to manage the input-output operations. In the disclosed technology, one physical zone represents the logical older copy of the data, whereas the other physical zone represents the logical newer copy of the data. Once the zone has been fully written, the disclosed technology detaches the older physical zone from the logical zone and erasure of the data present in the older physical zone is scheduled so that the older physical zone can be used to perform other input-output operations. Additionally, the disclosed technology uses a zone random write area (or a buffer space) that can be used as a staging buffer to manage random input-output operations so that random input-output operations do not trigger the rewriting or erasing of the data in the access unit of the solid-state drive. By using the techniques illustrated above, the disclosed technology can support random input-output operations on the solid-state drives without significant operational delay.
[0066] Having thus described the basic concept of the technology, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the technology. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed

processes to any order except as may be specified in the claims. Accordingly, the technology is limited only by the following claims and equivalents thereto.

Claims

1. A method implemented by a computing device and comprising: identifying, in response to a received write operation, a first physical zone and a second physical zone that are within a zoned namespace solid-state drive and mapped to a logical zone; temporarily staging data to be written as a result of the received write operation in a zone random write area associated with the second physical zone, wherein the temporarily staged data includes missing data read from the first physical zone; and transferring the temporarily staged data to the second physical zone, when a storage threshold of the zone random write area is determined to have been reached, to perform a commit operation.
2. The method of claim 1, wherein the first physical zone comprises a copy of an older version of other data and the second physical zone comprises another copy of a newer version of the other data.
3. The method of claim 2, further comprising erasing the older version of the other data from the first physical zone, when an end of the second physical zone is determined to have been reached subsequent to the commit operation, in order to make the first physical zone available for reuse.
4. The method of claim 1, further comprising destaging the zone random write area after transferring the temporarily staged data.
5. The method of claim 1, further comprising servicing a request to read additional data from the first physical zone, when the request to read the additional data is referenced to a location beyond a current location of the zone random write area.
6. The method of claim 1, further comprising servicing a request to read additional data from the second physical zone, when the request to read the additional data is referenced to a location before a current location of the zone random write area.
7. The method of claim 1, further comprising transferring the temporarily staged data to the second physical zone upon receipt of a commit command.
8. A non-transitory machine-readable medium having stored thereon instructions comprising executable code that, when executed by at least one computing device, causes the computing device to: identify, in response to a received write operation, a first physical zone and a second physical zone that are within a zoned namespace solid-state drive and mapped to a logical zone, wherein the first physical zone comprises a copy of an older version of data and the second physical zone comprises another copy of a newer version of the data; temporarily stage other data to be written as a result of the received write operation in a zone random write area associated with the second physical zone, wherein the temporarily staged other data includes missing data read from the first physical zone; and transfer the temporarily staged other data to the second physical zone, when a storage threshold of the zone random write area is determined to have been reached, to perform a commit operation.
9. The non-transitory machine-readable medium as set forth in claim 8, wherein the executable code, when executed by the computing device, further causes the computing device to erase the older version of the data from the first physical zone, when an end of the second physical zone is determined to have been reached subsequent to the commit operation, in order to make the first physical zone available for reuse.
10. The non-transitory machine-readable medium as set forth in claim 8, wherein the executable code, when executed by the computing device, further causes the computing device to destage the zone random write area after transferring the temporarily staged other data.
11. The non-transitory machine-readable medium as set forth in claim 8, wherein the executable code, when executed by the computing device, further causes the computing device to service a

request to read additional data from the first physical zone, when the request to read the additional data is referenced to a location beyond a current location of the zone random write area.
12. The non-transitory machine-readable medium as set forth in claim 8, wherein the executable code, when executed by the computing device, further causes the computing device to service a request to read additional data from the second physical zone, when the request to read the additional data is referenced to a location before a current location of the zone random write area.
13. The non-transitory machine-readable medium as set forth in claim 8, wherein the executable code, when executed by the computing device, further causes the computing device to transfer the temporarily staged other data to the second physical zone upon receipt of a commit command.
14. A computing device, comprising memory having instructions stored thereon and one or more processors coupled to the memory and configured to execute the stored instructions to: temporarily stage data to be written in a zone random write area associated with a second physical zone, wherein the temporarily staged data includes missing data read from a first physical zone; and perform a commit operation by transferring the temporarily staged data to the second physical zone, when a storage threshold of the zone random write area is determined to have been reached, wherein the first physical zone and the second physical zone are within a zoned namespace solid-state drive and mapped to a logical zone.
15. The computing device as set forth in claim 14, wherein the first physical zone comprises a copy of an older version of other data and the second physical zone comprises another copy of a newer version of the other data.
16. The computing device as set forth in claim 15, wherein the one or more processors are further configured to execute the instructions to erase the older version of the other data from the first physical zone, when an end of the second physical zone is determined to have been reached subsequent to the commit operation, in order to make the first physical zone available for reuse.
17. The computing device as set forth in claim 14, wherein the one or more processors are further configured to execute the instructions to destage the zone random write area after transferring the temporarily staged data.
18. The computing device as set forth in claim 14, wherein the one or more processors are further configured to execute the instructions to service a request to read additional data from the first physical zone, when the request to read the additional data is referenced to a location beyond a current location of the zone random write area.
19. The computing device as set forth in claim 14, wherein the one or more processors are further configured to execute the instructions to service a request to read additional data from the second physical zone, when the request to read the additional data is referenced to a location before a current location of the zone random write area.
20. The computing device as set forth in claim 14, wherein the one or more processors are further configured to execute the instructions to transfer the temporarily staged data to the second physical zone upon receipt of a commit command.