(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 March 2016 (03.03.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/032955 A2

(51) International Patent Classification:
G11C 8/06 (2006.01)

(21) International Application Number:
PCT/US2015/046534

(22) International Filing Date:
24 August 2015 (24.08.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/041,318 25 August 2014 (25.08.2014) US

(71) Applicant: **CACHEIO LLC** [US/US]; 330 Valley Road, Mason, New Hampshire 03048 (US).

(72) Inventors: **MANN, Bruce Eric**; 330 Valley Road, Mason, New Hampshire 03048 (US). **CROSS, Matthew Edward**; 355 Merriam Hill Rd, Mason, New Hampshire 03048 (US). **BEAVERSON, Arthur James**; 37 Cobleigh Rd, Boxborough, Massachusetts 01719 (US). **CHANG, Bang**; 101 Meadowstone Court, Cary, North Carolina 27513 (US).

(74) Agent: **GREEN, Edward, H., III**; Coats & Bennett, PLLC, 1400 Crescent Green, Suite 300, Cary, NC 27518 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

[Continued on next page]

(54) Title: NVRAM ENABLED STORAGE SYSTEMS

(57) Abstract: A method of preventing data loss is disclosed that includes writing data first to a NVRAM device, then to a storage device with volatile memory and persistent medium, and checking whether data is on the persistent medium before deleting the data from the NVRAM. A method of reducing read-modify-writes and write amplification is also disclosed that includes writing data first to a NVRAM device, accumulating data in full stripes before writing the data to a RAID system in full stripes and to each storage device in contiguous chunks sequentially.

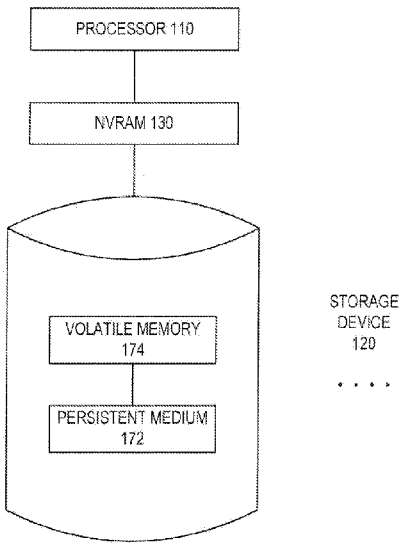


FIG. 2

WO 2016/032955 A2

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

NVRAM ENABLED STORAGE SYSTEMS

PRIORITY CLAIM

The present application claims priority to US Provisional Application No. 62/041,318 filed on August 25, 2014.

TECHNICAL FIELD

The present disclosure relates generally to storage systems, and, more specifically, to non-volatile random access memory (NVRAM) enabled storage systems.

BACKGROUND

Solid-state devices (SSDs), including solid-state drives and flash memory cards, are becoming increasingly popular for deployment in Redundant Array of Independent Disks (RAID) systems to support transactional applications such as databases. The RAID system may take the form of a hardware RAID card, RAID on a Chip, software RAID, Erasure Coding, or JBOD (Just a Bunch of Disks). Transactional applications typically issue read and write requests (I/O requests) that have small transfer sizes and are not in sequential block address order (collectively referred to as "random" I/O requests). SSDs typically service random read requests many times faster than traditional hard disk drives (HDDs). However, when SSDs are deployed in traditional RAID-5 and RAID-6 configurations, the random write performance is limited by the need to perform read-modify-write operations. The RAID system handles each random write request by reading existing data and parity, calculating a new parity, and then writing the new data and the new parity. These read-modify-write operations significantly reduce write performance and SSD endurance.

SSD write amplification also reduces write performance and SSD endurance. An SSD is comprised of a plurality of flash pages. An entire flash page must be “erased” before it can be rewritten. There is a write cycle limit to how many times a flash page can be erased and rewritten. When a transactional application writes to an SSD in a RAID system, its write request size will likely be much smaller than the SSD’s flash page size, resulting in partially written flash pages. Consequently the SSD has to perform garbage collection by moving user data from one partially written flash page to another until an entire flash page contains no more user data and can be erased. Garbage collection turns each application write into multiple SSD writes, also known as write amplification. Given the write cycle limit on each flash page, write amplification

significantly reduces SSD endurance and write performance. When an application or a storage system writes to an SSD in multiple small transfer sizes in sequential block address order (sequential writes), the SSD typically can fill entire flash pages with fewer partially written pages, reducing the write amplification during garbage collection.

An SSD typically comprises a persistent flash medium for storing data and a volatile memory to hold data temporarily before the data is committed to the persistent flash medium. In the event of a power failure, the data stored in the volatile memory will be lost. To prevent data loss during power failure, some of the SSDs are equipped with a capacitor or battery, which provides enough power for flushing the data stored in the volatile memory to the persistent flash medium. However, the additional capacitor can significantly increase the cost of the SSDs.

There is a need to design a storage system that reduces read-modify-write operations, minimizes SSD write amplification, and prevents data loss without the expensive capacitors.

SUMMARY

Methods and apparatus for improving data storage systems are disclosed.

In some embodiments, a storage system is configured to prevent data loss in the event of power failure. The storage system comprises a processor, one or more storage devices and a non-volatile memory (NVRAM). Each of the one or more storage devices comprises a persistent medium. The NVRAM device is configured to store one or more data blocks to be sent to a storage device for persistent storage. The processor is configured to check whether a data block stored on the NVRAM is also stored on the storage device's persistent medium before deleting the data block from the NVRAM.

In some embodiments, a storage system is configured to reduce read-modify-write operations and write amplification. The storage system comprises a processor, a RAID system with one or more storage devices, a NVRAM device and a memory. The NVRAM device stores one or more data blocks that are to be sent to the RAID system for persistent storage. The memory stores a metadata that maps every data block's logical block address (LBA) to its physical block address (PBA). The processor is configured to handle random write requests from an application. When handling random write requests, the processor first stores a data block in the NVRAM, then acknowledges to the application that the write request has been committed. Only after one or more data blocks in the NVRAM have been accumulated into a full RAID stripe, the full RAID stripe is written to the RAID system to reduce read-modify-write

operations. After the full RAID stripe has been written to the RAID system, the metadata is updated to map the LBAs of the one or more data blocks to the PBAs of the one or more data blocks. The one or more data blocks are deleted from the NVRAM after the metadata has been updated.

Of course, the present invention is not limited to the features, advantages, and contexts summarized above, and those familiar with storage technologies will recognize additional features and advantages upon reading the following detailed description and upon viewing the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a block diagram of a storage system with NVRAM devices.

Figure 2 illustrates a block diagram of a storage device with volatile memory and persistent medium.

Figure 3 illustrates the deferred write process at the NVRAM.

Figure 4 illustrates the deferred write process at the storage device.

Figure 5 illustrates a flow diagram of the check block persistent process.

Figure 6 illustrates a flow diagram of the make block persistent process.

Figure 7 illustrates a block diagram of a storage system with a RAID system.

Figure 8 illustrates a block diagram of RAID system data layout.

Figure 9 illustrates a flow diagram of NVRAM enabled writes to a RAID system.

Figure 10 illustrates a flow diagram of NVRAM enabled metadata updates.

DETAILED DESCRIPTION

A storage system is disclosed that has at least one NVRAM device to accomplish (1) preventing data loss in the event of a power failure; (2) reducing read-write-modify operations; and (3) reducing solid-state device write amplification.

Fig. 1 illustrates one embodiment of a storage system 100 that includes a processor 110 and one or more storage devices 120. Examples of storage device include solid-state device (SSD), hard disk drive (HDD), and a combination of SSDs and HDDs (Hybrid). The storage system 100 provides persistent storage to one or more user applications 140. In some embodiments, there may be multiple storage systems 100 implemented as a cluster for redundancy and performance. In some embodiments the storage device 120 may be accessible by multiple storage systems 100 as shared storage device. In some embodiments the application 140 and the storage system 100 may be running on the same physical system. In other embodiments the application 140

may access the storage system through a storage network such as FibreChannel, Ethernet, InfiniBand, and PCIe.

The processor 110 interfaces between the application 140 and the storage device 120. The processor 110 controls and manages the storage device 120. For example, the processor 110 may provide a set of commands for the application 140 to read from and write to the storage device 120. Also the processor 110 can provide redundancy, performance, and data services that often can't be achieved by the storage device 120.

In the present disclosure the storage system 100 includes one or more non-volatile random-access memory (NVRAM) devices 130. Examples of NVRAM include battery-backed DRAM, NVDIMM, PCIe NVRAM card, and solid-state device. In some embodiments, upon receiving a write request from the application 140, the processor 110 stores the write data in the NVRAM 130 and acknowledges to the application 140 that the write request is successful before the data is actually committed to the storage device 120. This process is known as deferred write.

Fig. 2 illustrates one embodiment of a storage device 120, such as a solid-state device (SSD), that comprises a persistent medium 172 for storing data and a volatile memory 174 for buffering data temporarily before the data is committed to the persistent medium 172. In the event of a power failure, the data stored in the volatile memory 174 will be lost. Traditionally, to prevent data loss during power failure, an SSD is equipped with a capacitor or battery, which provides enough power to write all the data in the volatile memory 174 to the persistent medium 172. However, the capacitor or battery can significantly increase the cost of the SSD.

The present disclosure provides methods for preventing data loss during power failure without the additional capacitor or battery in the SSD.

Fig. 3 illustrates the deferred write process at the NVRAM 130:

Step 1: The processor 110 receives a write request from the application 140;

Step 2: The processor 110 commits the write data to the NVRAM 130;

Step 3: The processor 110 acknowledges to the application 140 that the write is successful;

Step 4: At a later time the processor 110 writes the data in the NVRAM 130 to the storage device 120 (deferred write);

Step 5: At a later time the processor 110 deletes the data from the NVRAM 130 so the NVRAM space can be reused.

Fig. 4 illustrates the deferred write process at the storage device 120:

Step 4': The storage device 120 receives a write request from the processor;

Step 6: The storage device 120 stores the write data in its volatile memory 174;

Step 7: The storage device 120 acknowledges to the processor 110 that the write is successful;

Step 8: At a later time the storage device 120 writes the data in the volatile memory 174 to its persistent medium 172

If a power failure takes place after step 5 but before step 8, the write data will be lost. In order to prevent data loss the present disclosure replaces step 5 with the following steps as illustrated in Fig. 5:

Step 510: The processor 110 selects a data block in the NVRAM 130 to be deleted so its NVRAM space can be reused;

Step 520: The processor 110 checks if the data block is on the storage device's persistent medium 172 by issuing a "check block persistent" request to the storage device 120;

Step 530: If the storage device responds "yes", the processor 110 deletes the data block from the NVRAM 130.

Step 540: If the storage device responds "no", the processor 110 issues a "flush block" request to the storage device 120. Upon receiving the request, the storage device 120 writes the data block from its volatile memory 174 to its persistent medium 172 and acknowledges completion to the processor 110;

Step 550: Upon receiving the acknowledgement the processor 110 deletes the data block from the NVRAM 130.

Following the above steps ensures that in the event of a power failure the data can be recovered from either the NVRAM 130 or the storage device's persistent medium 172.

Fig. 6 illustrates another embodiment of the present disclosure by replacing step 5 with the following steps:

Step 610: The processor 110 selects a data block in the NVRAM 130 to be deleted so its space can be reused;

Step 620: The processor 110 issues "make block persistent" request to the storage device 120. Upon receiving the request, the storage device 120 checks if the data block is on its persistent medium. If not the storage device 120 writes the data block from its volatile memory 174 to its persistent medium 172. The storage device then acknowledges completion to the processor 110.

Step 630: The processor 110 receives the completion acknowledgement for "make block persistent" request;

Step 640: The processor 110 deletes the data block from the NVRAM 130.

Following the above steps ensures that in the event of a power failure the data can be recovered from either the NVRAM 130 or the storage device's persistent medium 172.

In some embodiments the NVRAM device 130 is configured to be much larger than the aggregate size of the volatile memory in the storage devices 120. This allows the processor 110 to delay deleting a data from the NVRAM as much as possible so the data is more likely to have been flushed to the persistent medium 172 before a check block consistent or make block persistent request. In some embodiments the storage device 120 always writes data from its volatile memory 174 to its persistent medium 172 first in first out (FIFO). In these embodiments the processor may calculate whether a data is on the persistent medium without the check block persistent or make block persistent request.

Fig. 7 illustrates one embodiment of a storage system 100 that includes a RAID (Redundant Array of Independent Disks) system 150 between the processor 110 and the storage device 120. Examples of RAID system include software RAID, hardware RAID card, RAID on a chip, Erasure Coding, or JBOD (Just a Bunch of Disks). The RAID system may be configured in write through (WT) or write back (WB) mode. The RAID system 150 virtualizes multiple storage devices 120 into logical units. The RAID system 150 may be implemented to distribute data blocks across the storage devices 120 (i.e., striping) and generate checksums (i.e., parity) for data redundancy and recovery. However, under transactional workload the RAID system introduces read-modify-write operations that reduce write performance. In some embodiments the storage devices 120 are SSDs. Small transactional writes to a SSD causes write amplification, which reduces SSD endurance and write performance.

The present disclosure provides methods for reducing read-modify-write operations and SSD write amplification. In one embodiment the processor 110 maintains Metadata 160 that maps every data block's LBA (Logical Block Address) to its PBA (Physical Block Address). LBA is the virtual block address assigned and accessed by the application 140 whereas PBA represents the block's physical location in the RAID system 150. In some embodiments the metadata 160 may map every data block's LBA to its content ID (e.g. content fingerprint) and every content ID to its PBA.

Fig. 8 illustrates one embodiment wherein the storage devices 120 in the RAID system 150 are managed as one or more Beads 186. Each Bead comprises one or more contiguous RAID stripes 184. Each RAID stripe comprises one chunk 182 from each storage device. The processor 110 is configured to write to fill one or more Beads (current Beads) before writing to new Beads. Fig. 9 illustrates the write data flow:

Step 710: Upon receiving an application write request, the processor 110 commits the write data in the NVRAM 130 and acknowledges completion to the application 140. The processor 110 accumulates one or more data blocks in the NVRAM 130 into a full RAID stripe 184;

Step 720: The processor 110 checks if the current Bead is filled;

Step 730: If not the processor 110 writes the full RAID stripe in one or more transfers after the existing RAID stripes in the current Bead;

Step 740: If yes the processor 110 writes the RAID stripe in one or more transfers at the beginning of a new Bead;

Step 750: The processor 110 updates the metadata 160 to map the LBA of each data block in the RAID stripe to its PBA;

Step 760: At a later time the processor 110 deletes the data blocks from the NVRAM so their NVRAM space can be reused

The above write data flow ensures that the RAID system 150 receives mostly full stripe write (FSW) requests, which cause fewer or no read-modify-write operations. It also ensures that most data is written to each SSD in contiguous chunks (sequential writes), which reduces the SSD's write amplification.

In some embodiments the processor 110 makes metadata 160 persistent by writing metadata updates to the RAID system 150. Metadata updates are typically of small transfer sizes and are another source for read-modify-writes and SSD write amplification. Fig. 10 illustrates the metadata update data flow for reducing read-modify-writes and write amplification:

Step 810: The processor 110 commits metadata update in the NVRAM 130;

Step 820: The processor 110 accumulates one or more metadata updates in the NVRAM into one RAID stripe 184;

Step 830: The processor 110 writes the RAID stripe in one or more transfers to the current Bead or a new Bead;

Step 840: The processor 110 updates the metadata index to the PBA of the on-disk metadata structure;

Step 850: The processor 110 deletes the metadata updates from the NVRAM 130 so their NVRAM space can be reused.

In some embodiments the metadata updates have their own Beads separate from data Beads. In other embodiments the metadata updates are mixed with data in the same Beads.

The foregoing description and the accompanying drawings represent non-limiting examples of the methods and apparatus taught herein. As such, the present disclosure is not limited by the foregoing description and accompanying drawings. Instead, the present disclosure is limited only by the following claims and their legal equivalents.

CLAIMS

What is claimed is:

1. A storage system configured to prevent data loss in the event of power failure, the storage system comprising:

a processor;

one or more storage devices, each of the one or more storage devices comprising a volatile memory and a persistent medium; and

a non-volatile memory (NVRAM) device configured to store one or more data blocks to be sent to a storage device for persistent storage;

wherein the processor is characterized by checking whether a data block stored on the NVRAM device is also stored on the storage device's persistent medium before deleting the data block from the NVRAM.

2. The storage system of claim 1, wherein the processor is configured to request that the storage device make the data block persistent if the data block is not stored on its persistent medium.

3. The storage system of claim 1, wherein the processor is configured to make a data block persistent before deleting it from the NVRAM by issuing a "make block persistent" request to the storage device, wherein the storage device handles the "make block persistent" request by:

checking whether the data block is stored in the storage device's volatile memory;

if the data block is stored in its volatile memory, transferring the data block to its persistent medium; and

acknowledging to the processor that the data block is stored on the persistent medium.

4. The storage system of claim 1, wherein the NVRAM is a battery-backed DRAM, NVDIMM, NVRAM card, solid-state device, or other non-volatile memory device.

5. The storage system of claim 1, wherein the NVRAM is mirrored to one or more storage systems for redundancy.

6. The storage system of claim 1, wherein the one or more storage devices are solid-state devices (SSDs), traditional hard disk drives (HDDs), a combination of SSD and HDDs, or other persistent medium devices.

7. The storage system of claim 1, wherein the one or more storage devices are deployed in a Redundant Array of Inexpensive Disks (RAID) system; wherein the RAID system takes the form of a software RAID, hardware RAID card, RAID on a chip, or other RAID technologies.

8. The storage system of claim 1, wherein the one or more storage devices are deployed in an erasure coding system; wherein the erasure coding system takes the form of a software, hardware card, a chip, or other erasure coding technologies.

9. The storage system of claim 1, wherein the one or more storage devices are deployed in a Just-a-Bunch-of-Disks (JBOD) system; wherein the JBOD system takes the form of a software, hardware card, a chip, or other JBOD technologies.

10. The storage system of claim 1, where the storage devices can be accessed from one or more storage systems.

11. A storage system configured to reduce read-modify-write operations and write amplifications, the storage system comprising:

a processor; and

a Redundant Array Inexpensive Disk (RAID) system with one or more storage devices; and

a non-volatile-memory (NVRAM) device, wherein the NVRAM device stores one or more data blocks to be sent to the RAID system for persistent storage;

a memory that stores a metadata, wherein the metadata maps the one or more data blocks' logical block addresses (LBAs) to their physical block addresses (PBAs);

wherein the processor is configured to handle random write requests from an application by:

storing a data block in the NVRAM device;

acknowledging to the application that the write request has been committed;

accumulating one or more data blocks in the NVRAM into a full RAID stripe;

writing the full RAID stripe to the RAID system to reduce read-modify-write operations;

updating the metadata to map the LBAs of the one or more data blocks to the PBAs of the one or more data blocks;

deleting the one or more data blocks from the NVRAM device.

12. The storage system of claim 11, wherein the processor is configured to make the metadata updates persistent by:

storing each metadata update in the NVRAM device;

accumulating one or more metadata updates with or without data blocks in the NVRAM device into full RAID stripes;

writing full RAID stripes to the RAID system to reduce read-modify-write operations;

updating metadata index with on-disk metadata PBAs; and

deleting the metadata updates from the NVRAM device so the NVRAM device.

13. The storage system of claim 11, wherein the RAID system is operating in a write-through (WT) mode.

14. The storage system of claim 11, wherein one or more data and metadata blocks are written to each storage device in contiguous chunks sequentially to reduce the storage device's write amplification.

15. The storage system of claim 11, wherein the NVRAM is a battery-backed DRAM, NVDIMM, NVRAM card, solid-state device, or other non-volatile memory device.

16. The storage system of claim 11, wherein the NVRAM is mirrored to one or more storage systems for redundancy.

17. The storage system of claim 11, wherein the one or more storage devices are solid-state devices (SSDs), traditional hard disk drives (HDDs), a combination of SSDs and HDDs (hybrid), or other persistent medium devices.

18. The storage system of claim 11, wherein the one or more storage devices are deployed in a RAID (Redundant Array of Inexpensive Disks) system; wherein the RAID system takes the form of a software RAID, hardware RAID card, RAID on a chip, or other RAID technologies.

19. The storage system of claim 11, wherein the one or more storage devices are deployed in an erasure coding system; wherein the erasure coding system takes the form of a software, hardware card, a chip, or other erasure coding technologies.

20. The storage system of claim 11, where the one or more storage devices are deployed in a Just-a-Bunch-of-Disks (JBOD) system; wherein the JBOD system takes the form of a software, hardware card, a chip, or other JBOD technologies.

21. The storage system of claim 11, where the storage devices can be accessed from one or more storage systems.

22. A method implemented in a storage system for preventing data loss, said storage system comprising a processor, one or more storage devices, and a non-volatile memory (NVRAM) device, said one or more storage devices comprising a persistent medium and a volatile memory, said method is characterized by:

sending data stored in the NVRAM device to the one or more storage devices; and

checking whether a data block stored on the NVRAM device is also stored on the storage device's persistent medium before deleting the data from the NVRAM.

23. The method of claim 22, further comprising:

requesting that the storage device make the data block persistent if the data block is not stored on the persistent medium.

24. The method of claim 22, further comprising:

issuing a “make persistent” request to the storage device to make a data block persistent before deleting it from the NVRAM; wherein the storage device handles the “make persistent request by:

checking whether the data block is stored in the storage device’s volatile memory;

if the data block is stored in its volatile memory, transferring the data block to its persistent medium; and

acknowledging to the processor that the data block is stored on the persistent medium.

25. A method implemented in a storage system for reducing read-modify-write operations and write amplifications, said storage system comprising a processor, a RAID system with one or more storage devices, and a non-volatile memory (NVRAM) device, and a memory that stores a metadata, said method comprising:

receiving a write request from an application to write a data block to a storage device;

storing the data block on the NVRAM device; and

acknowledging to the application that the write request has been committed;

wherein the method is further characterized by:

accumulating one or more data blocks in the NVRAM into a full RAID stripe;

writing the full RAID stripe to the RAID system to reduce read-modify-write operations;

updating the metadata to map the LBAs of the one or more data blocks to the PBAs of the one or more data blocks; and

deleting the one or more data blocks from the NVRAM device.

26. The method of claim 25, further comprising:

storing each metadata update in the NVRAM device;

accumulating one or more metadata updates with or without data blocks in the NVRAM device into full RAID stripes;

writing full RAID stripes to the RAID system to reduce read-modify-write operations;

updating metadata index with on-disk metadata PBAs; and

deleting the metadata updates from the NVRAM so that the NVRAM device can be used for new metadata updates.

27. The method of claim 25, wherein the RAID system is operating in a write-through (WT) mode.

28. The method of claim 25, wherein one or more data and metadata blocks are written to each storage device in contiguous chunks sequentially to reduce the storage device's write amplification.

29. A storage system configured to prevent data loss in the event of power failure, the storage system comprising:

a processor;

one or more storage devices, each of the one or more storage devices comprising a persistent medium and a volatile memory; and

a non-volatile memory (NVRAM) device configured to store one or more data blocks to be sent to a storage device for persistent storage;

wherein the NVRAM device is characterized by being configured to be larger than the aggregate size of the volatile memory, and wherein the processor is characterized by being configured to delay deleting a data from the NVRAM to allow the data to be flushed to the persistent medium before a check block consistent request or a make block persistent request is received.

30. The storage system of claim 29, wherein each of the one or more storage devices is configured to write data from the volatile memory to the persistent medium first in first out (FIFO) and wherein the processor is configured to calculate whether a data is on the persistent medium of a storage device without issuing the check block persistent or make block persistent request.

31. A method of preventing data loss in the event of power failure, said method being implemented in a storage system that comprises a processor, one or more storage devices, and a non-volatile memory (NVRAM) device, said method comprising:

configuring the NVRAM device to be larger than the aggregate size of the volatile memory;

storing one or more data blocks in the NVRAM device; and

sending the one or more data blocks stored in the NVRAM device to a storage device for persistent storage;

said method further characterized by:

delaying deleting a data from the NVRAM to allow the data to be flushed to the persistent medium before a check block consistent request or a make block persistent request is received.

32. The method of claim 31, wherein the one or more storage devices are configured to write data from the volatile memory to the persistent medium first in first out (FIFO), and wherein the processor is configured to calculate whether a data is on the persistent medium of a storage device without issuing the check block persistent or make block persistent request.

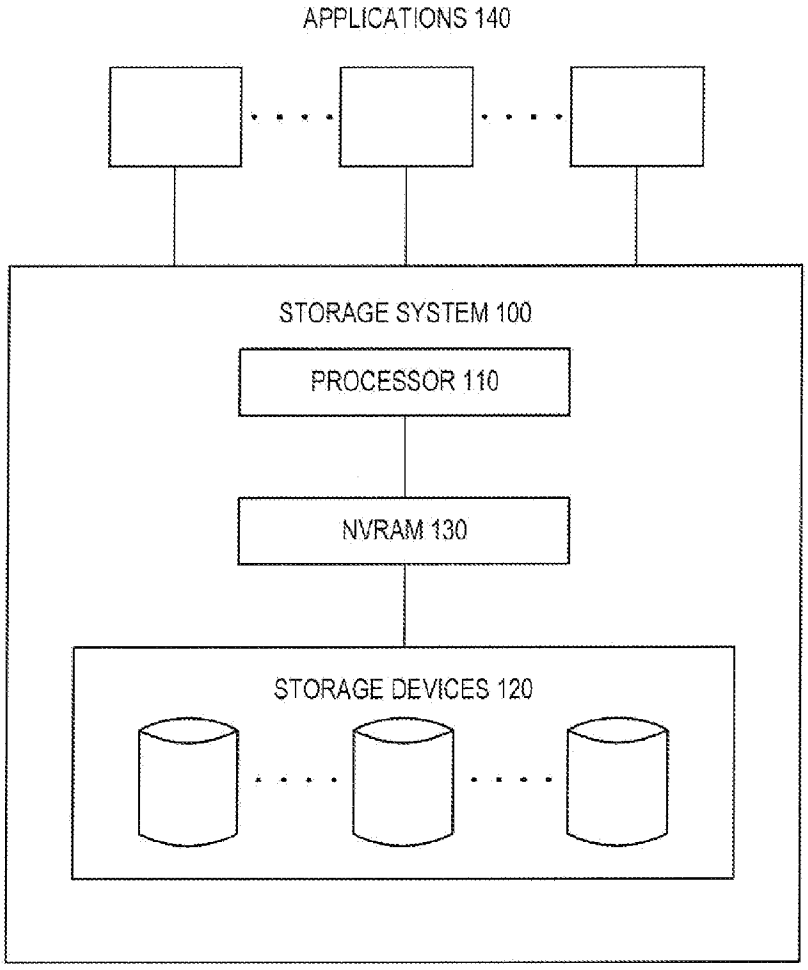

APPLICATIONS 140
STORAGE SYSTEM 100
PROCESSOR 110
NVRAM 130
STORAGE DEVICES 120

FIG. 1

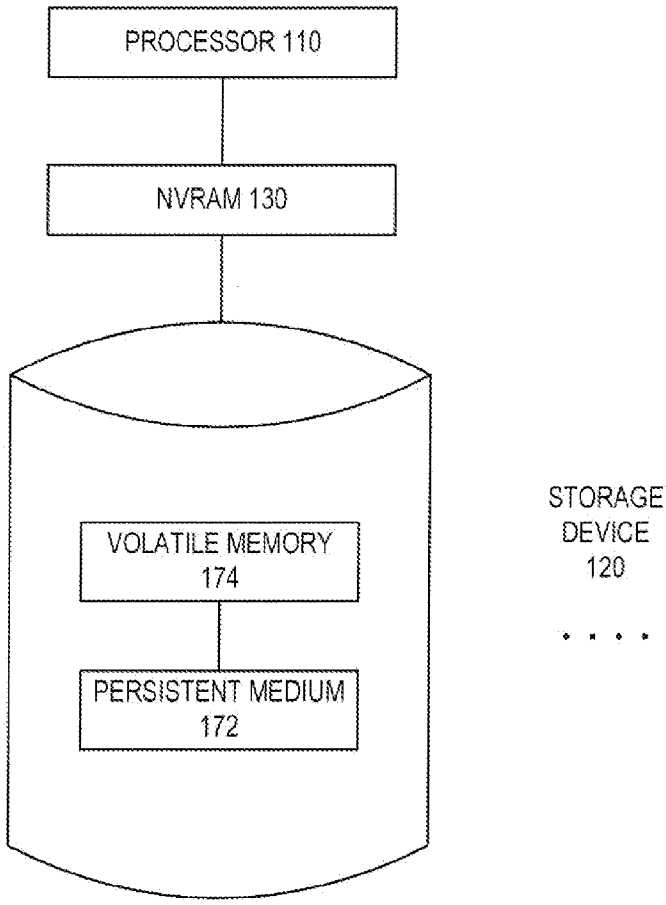
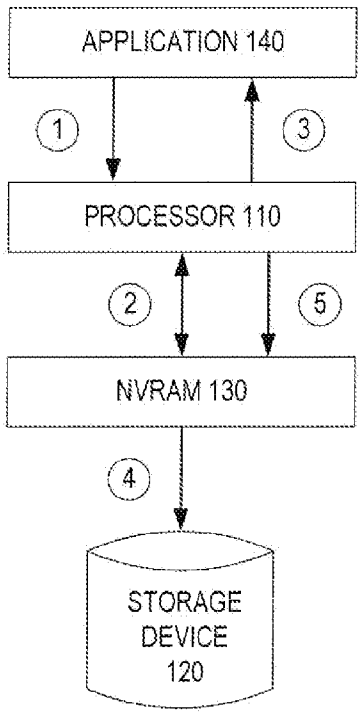

PROCESSOR 110
NVRAM 130
VOLATILE MEMORY 174
PERSISTENT MEDIUM 172
STORAGE DEVICE 120
. . . .

FIG. 2



1. RECEIVE A WRITE REQUEST FROM APPLICATION
2. COMMIT WRITE DATA TO NVRAM
3. ACKNOWLEDGE TO APPLICATION
4. DEFERRED WRITE TO STORAGE DEVICE
5. DELETE DATE FROM NVRAM

FIG. 3

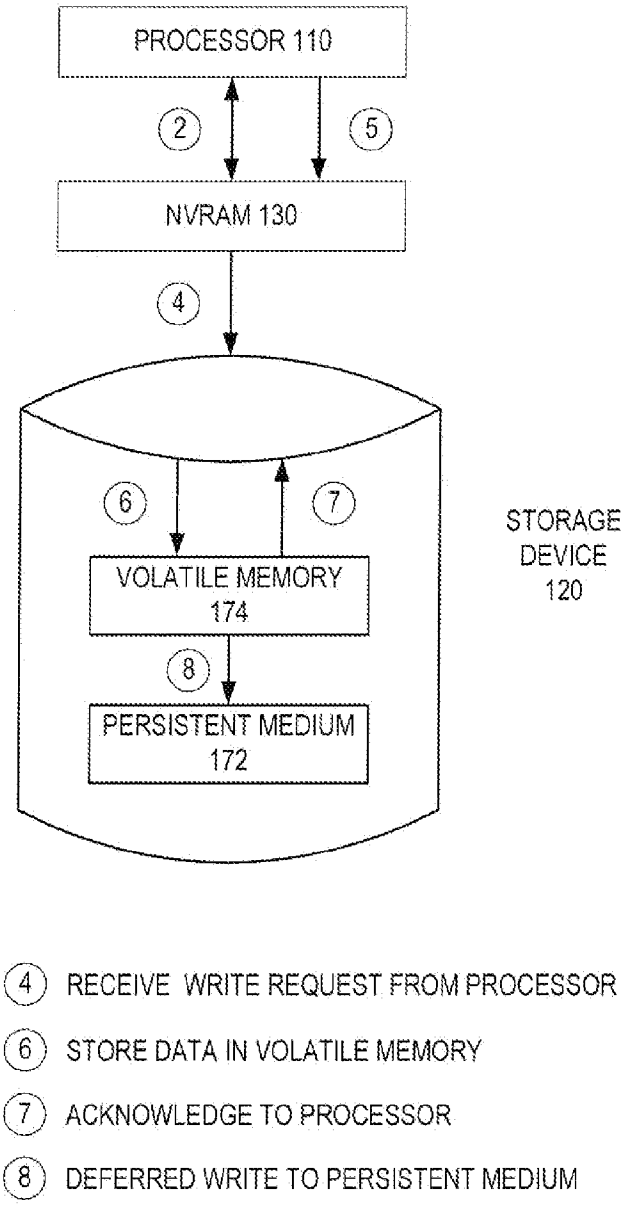

PROCESSOR 110
2
5
NVRAM 130
4
6
7
VOLATILE MEMORY 174
8
PERSISTENT MEDIUM 172
STORAGE DEVICE 120
4 RECEIVE WRITE REQUEST FROM PROCESSOR
6 STORE DATA IN VOLATILE MEMORY
7 ACKNOWLEDGE TO PROCESSOR
8 DEFERRED WRITE TO PERSISTENT MEDIUM

FIG. 4

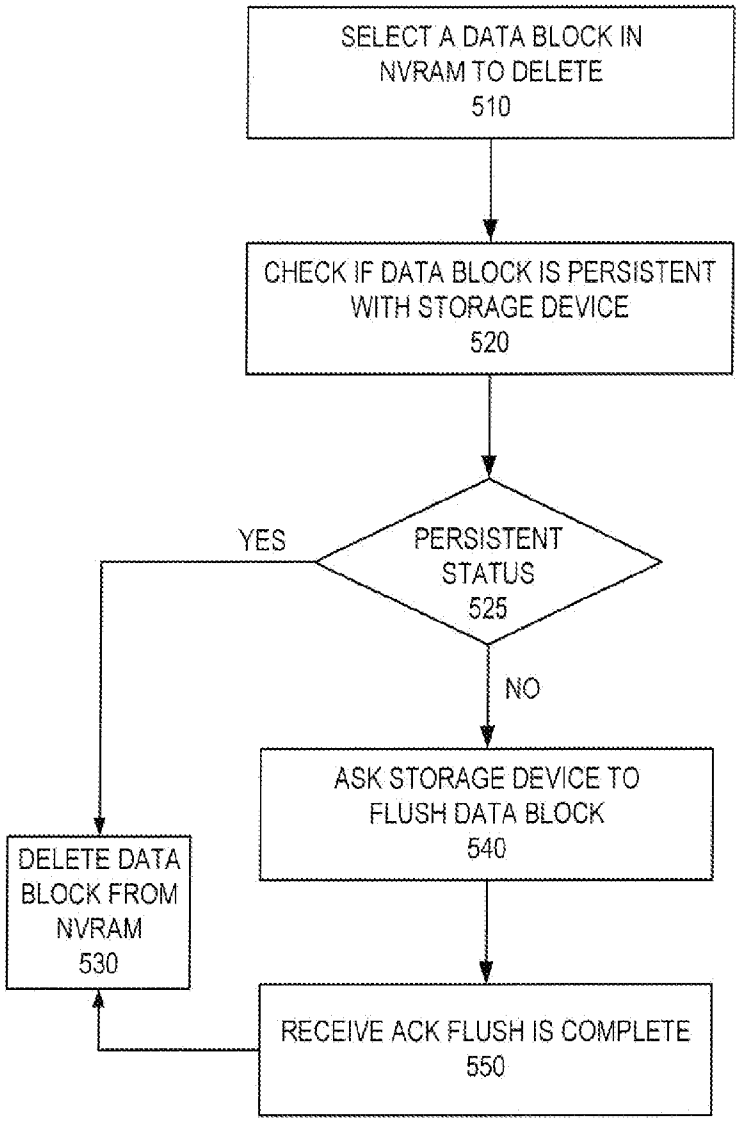

SELECT A DATA BLOCK IN NVRAM TO DELETE 510
CHECK IF DATA BLOCK IS PERSISTENT WITH STORAGE DEVICE 520
PERSISTENT STATUS 525
YES
NO
DELETE DATA BLOCK FROM NVRAM 530
ASK STORAGE DEVICE TO FLUSH DATA BLOCK 540
RECEIVE ACK FLUSH IS COMPLETE 550

FIG. 5

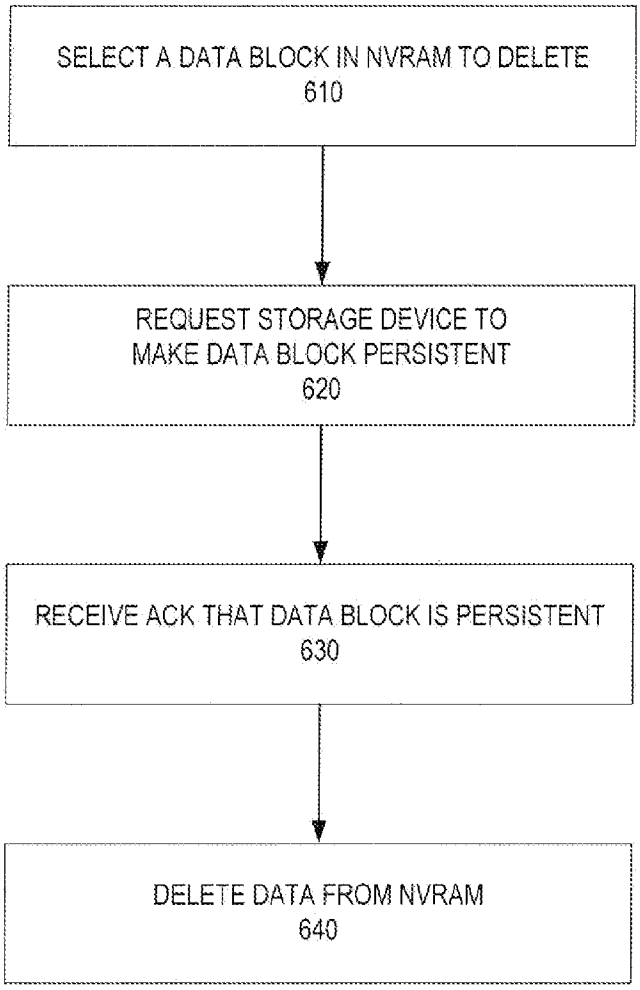

SELECT A DATA BLOCK IN NVRAM TO DELETE
610
REQUEST STORAGE DEVICE TO
MAKE DATA BLOCK PERSISTENT
620
RECEIVE ACK THAT DATA BLOCK IS PERSISTENT
630
DELETE DATA FROM NVRAM
640

FIG. 6

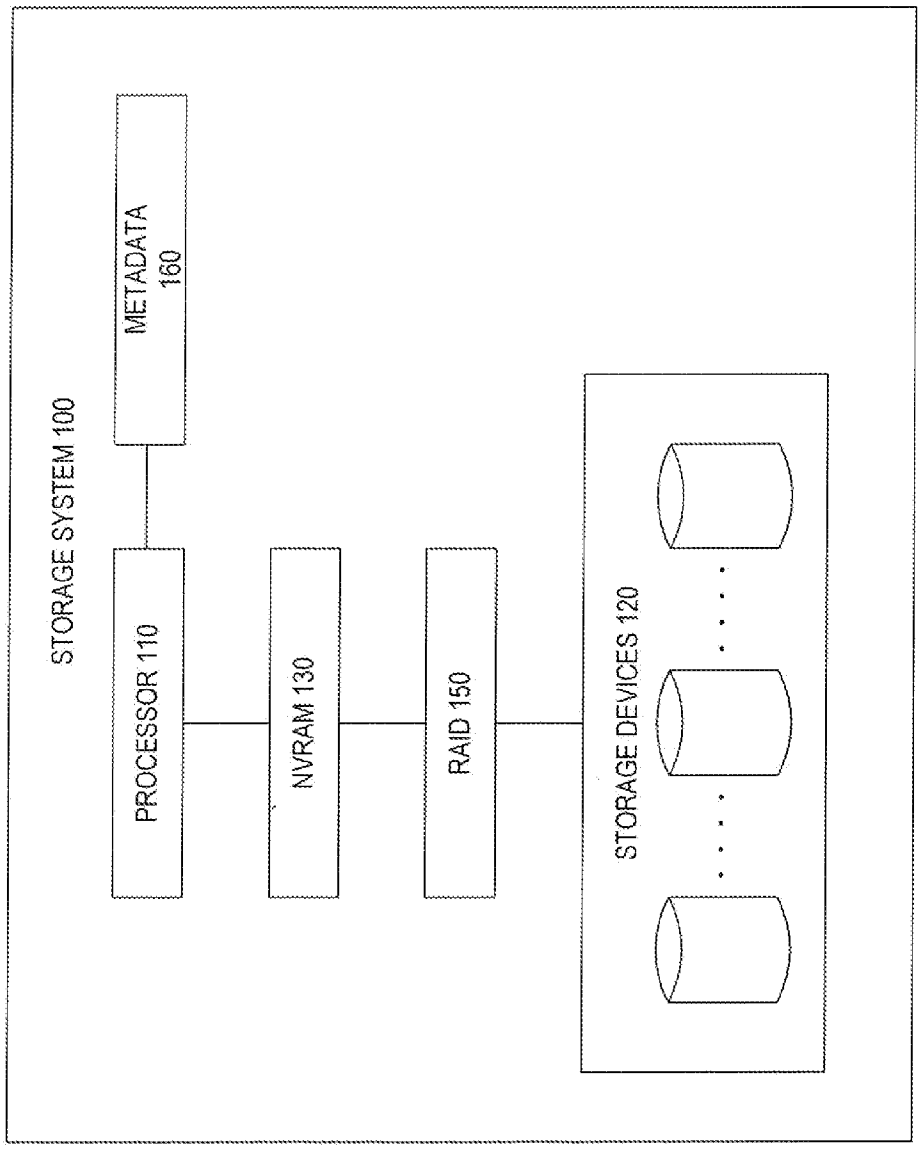

STORAGE SYSTEM 100
METADATA 160
PROCESSOR 110
NVRAM 130
RAID 150
STORAGE DEVICES 120

FIG. 7

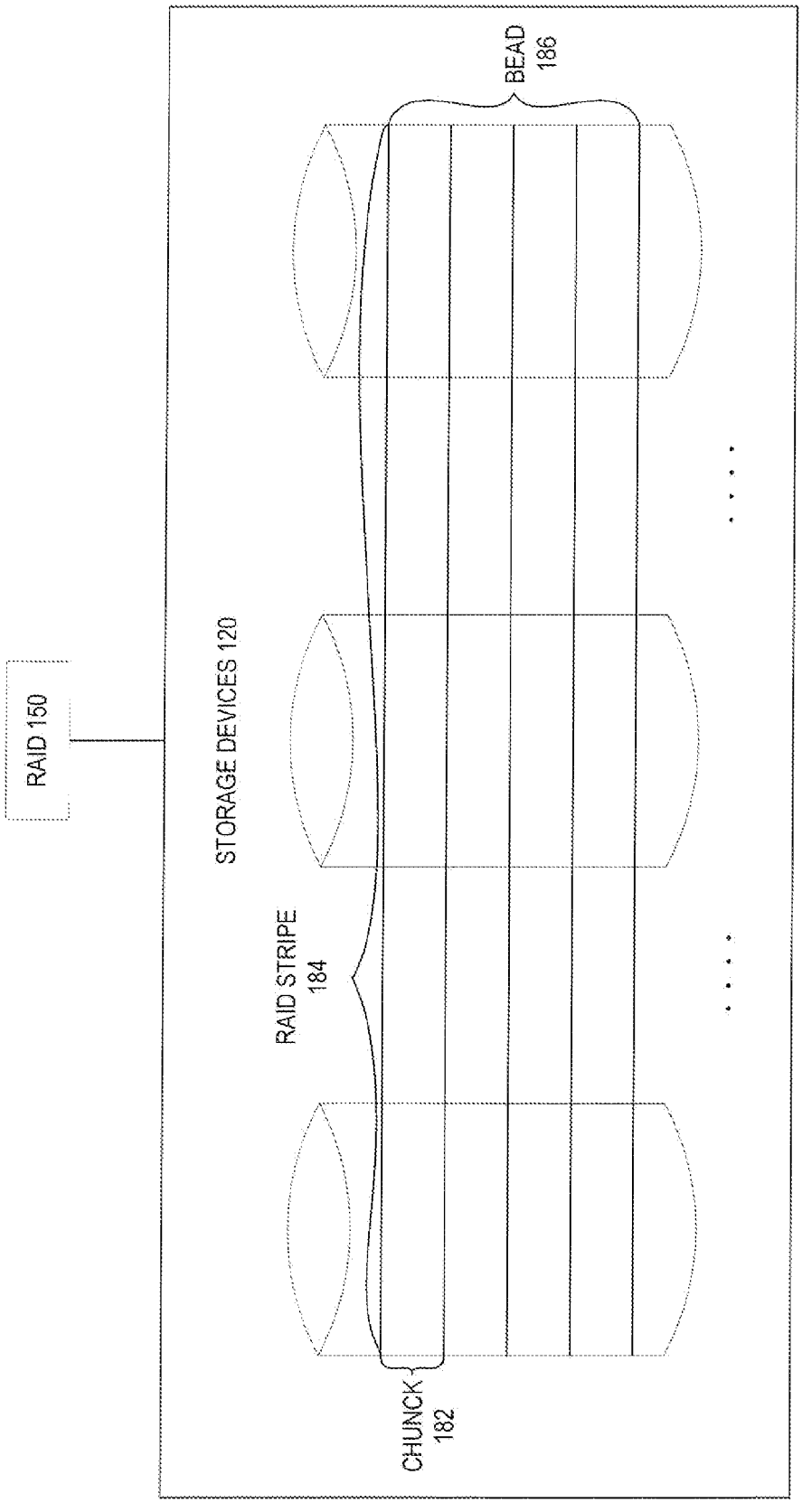

RAID 150
STORAGE DEVICES 120
BEAD
186
RAID STRIPE
184
CHUNCK
182

FIG. 8

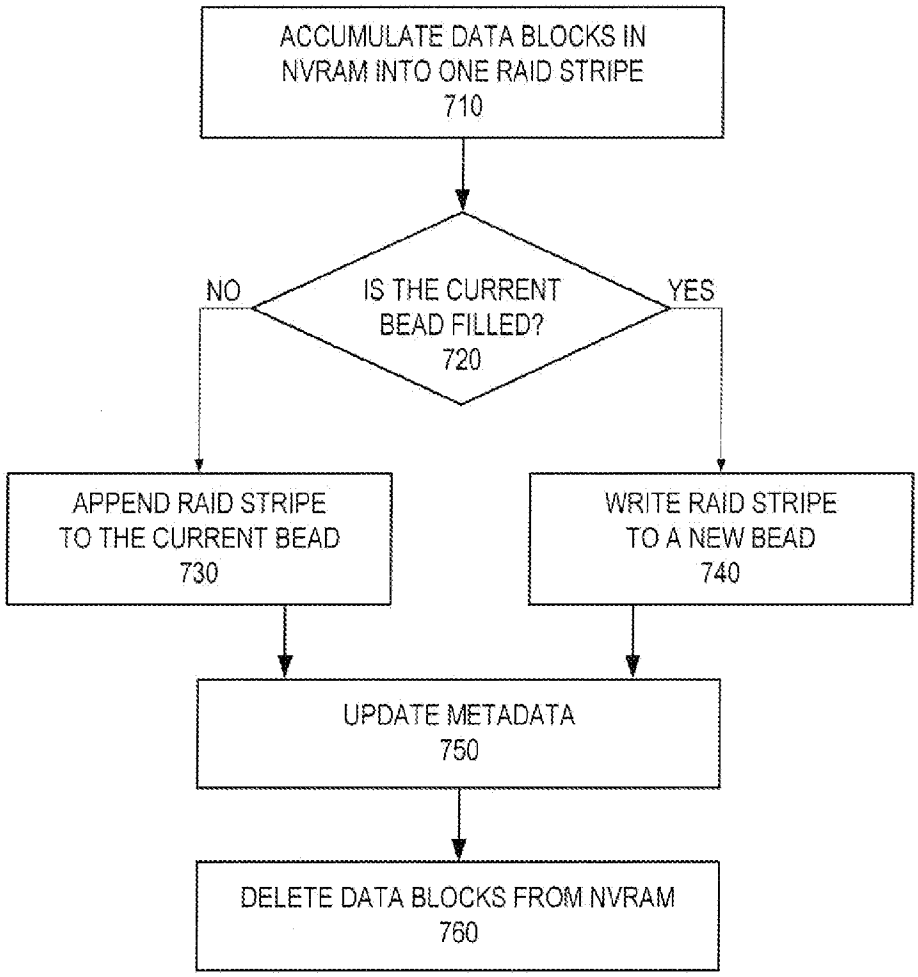

ACCUMULATE DATA BLOCKS IN NVRAM INTO ONE RAID STRIPE 710
IS THE CURRENT BEAD FILLED? 720
NO
YES
APPEND RAID STRIPE TO THE CURRENT BEAD 730
WRITE RAID STRIPE TO A NEW BEAD 740
UPDATE METADATA 750
DELETE DATA BLOCKS FROM NVRAM 760

FIG. 9

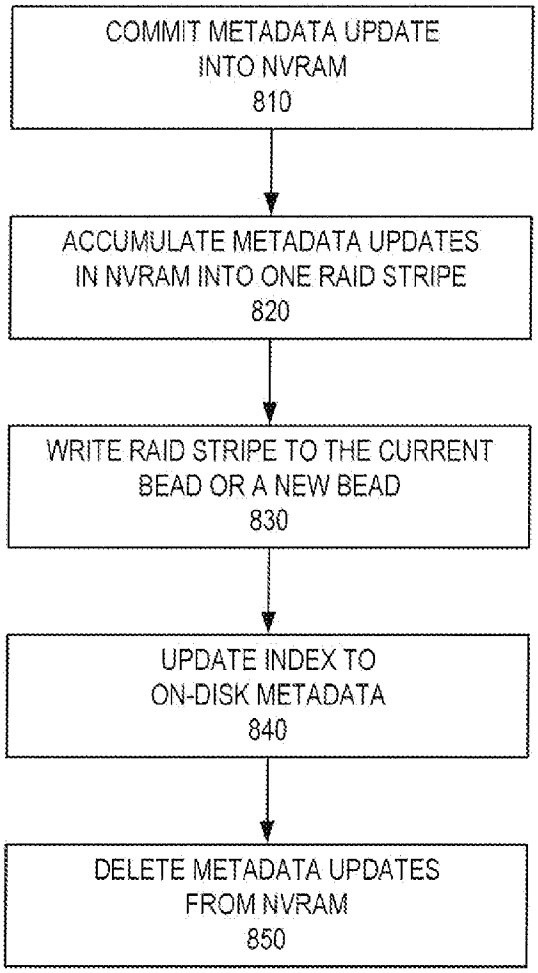

COMMIT METADATA UPDATE
INTO NVRAM
810
ACCUMULATE METADATA UPDATES
IN NVRAM INTO ONE RAID STRIPE
820
WRITE RAID STRIPE TO THE CURRENT
BEAD OR A NEW BEAD
830
UPDATE INDEX TO
ON-DISK METADATA
840
DELETE METADATA UPDATES
FROM NVRAM
850

FIG. 10