US Patent & Trademark Office
Patent Public Search | Text View

United States Patent	12386705
Kind Code	B2
Date of Patent	August 12, 2025
Inventor(s)	Pulickal Aravindakshan; Anoop

System and methods for computing parity information in a raid array

Abstract

A redundant disk array may include redundant information to facilitate rebuilding the array in the event of a disk failure. A host processor may allocate buffers in an accelerator memory. Data may be moved from one or more storage devices to the buffers. An accelerator engine may perform parity calculations required to rebuild the array based on data in the buffers without requiring a host CPU to perform the parity calculation.

Inventors:	**Pulickal Aravindakshan; Anoop (Karnataka, IN)**
Applicant:	**Microchip Technology Incorporated** (Chandler, AZ)
Family ID:	**1000008750773**
Assignee:	**Microchip Technology Incorporated (Chandler, AZ)**
Appl. No.:	**18/378414**
Filed:	**October 10, 2023**

Prior Publication Data

Document Identifier	Publication Date
US 20250045162 A1	Feb. 06, 2025

Foreign Application Priority Data

IN	202311052018	Aug. 02, 2023

Publication Classification

Int. Cl.: **G06F11/00** (20060101); **G06F11/10** (20060101); **G06F13/16** (20060101)

U.S. Cl.:

CPC **G06F11/1092** (20130101); **G06F11/1096** (20130101); **G06F13/1673** (20130101);

Field of Classification Search

CPC: G06F (11/1092); G06F (11/1096); G06F (13/1673); G06F (11/1076); G06F (13/28)

USPC: 714/1-57

References Cited

U.S. PATENT DOCUMENTS

Patent No.	Issued Date	Patentee Name	U.S. Cl.	CPC
6993680	12/2005	Fukumori	714/22	G06F 1/30
2011/0126045	12/2010	Bennett	714/6.22	G11B 20/1833
2013/0031420	12/2012	Haverkamp	714/45	G06F 11/3656
2013/0117603	12/2012	Jess	714/E11.041	G06F 11/1088
2013/0346793	12/2012	Flynn	714/6.3	G06F 12/0802
2016/0188424	12/2015	Walls	714/6.3	G06F 11/1662
2020/0319819	12/2019	Nelogal	N/A	G06F 11/1068
2021/0271547	12/2020	Bates	N/A	G11C 16/26
2024/0176622	12/2023	Ravi	N/A	G06F 9/3861

OTHER PUBLICATIONS

International Search Report and Written Opinion, Application No. PCT/US2024/014219, 11 pages, May 31, 2024. cited by applicant

Primary Examiner: Butler; Sarai E

Attorney, Agent or Firm: SLAYDEN GRUBERT BEARD PLLC

Background/Summary

PRIORITY
(1) This application claims priority to commonly owned Indian patent application No. 202311052018 filed on Aug. 2, 2023, the entire contents of which are hereby incorporated by reference for all purposes.
FIELD OF THE INVENTION
(2) The present disclosure relates to systems and methods for computing parity information in an array of storage devices.
BACKGROUND
(3) In high-density data storage applications, redundant data may be stored to protect against any loss of information. Additionally, a block of data may be written across multiple drives, in what are called strips, and parity information may be stored along with the data. In the event of a failure in one drive, the parity information may be used to recover any lost data.
(4) In one of various examples, a high-density storage application may be implemented as a Redundant Array of Independent Disks (RAID), such as RAID-5 without limitation. In one example of a RAID array, individual drives may be partitioned into strips, and consecutive

segments of data may be stored across different storage devices. In one of various examples, M individual drives, numbered 1 to M may, respectively, be partitioned into N strips, the strips numbered from 1 to N. Parity may be computed on a strip-wide basis across each of the M drives. In this example, to compute the parity information for the first strip, a parity calculation may be performed on the data in strip #1 for drives 1 to M−1, and the result stored in drive M.
(5) If a drive fails, the stored parity information may be used to calculate the missing data and prevent any loss of data. However, this calculation of the parity information requires significant computational overhead from the CPU.
(6) There is a need for a system and method to compute parity information in a RAID array that offloads the processing overhead from the CPU.
SUMMARY
(7) The examples herein enable a system and method to compute parity information in a RAID array that offloads the processing overhead from the CPU.
(8) According to one of various aspects, a system may include a central processing unit (CPU), a non-transitory processor memory device coupled to the CPU, one or more storage devices. The one or more storage devices may comprise a storage array and a Direct Memory Access (DMA) circuit. The system may include an accelerator processor, an accelerator non-transitory memory, a shared bus, where the shared bus is coupled to the CPU, the accelerator processor, the accelerator non-transitory memory, and the one or more storage devices. The CPU may instruct the one or more storage devices to move data between the one or more storage devices and the accelerator non-transitory memory. The CPU may instruct the accelerator processor to perform a parity calculation on data stored in the accelerator non-transitory memory and to write the parity information to the one or more storage devices.
(9) According to one of various aspects, a method may include operations of: allocating a source buffer and a result buffer in an accelerator memory in an accelerator device, moving data from one or more storage devices to the source buffer, issuing an interrupt from the one or more storage devices to a host processor, calculating parity information based on data in the source buffer, writing the parity information to the result buffer in the accelerator memory, and writing the parity information to the one or more storage devices.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
(1) FIG. **1** illustrates one of various examples of a RAID array device.
(2) FIG. **2** illustrates one of various examples of a system for computing parity information in a RAID array.
(3) FIG. **3** illustrates a method for computing parity information in a RAID array.
DETAILED DESCRIPTION
(4) FIG. **1** illustrates one of various examples of a RAID array device **100**. RAID array device **100** may be comprised of four storage devices **101**, **102**, **103**, and **104**. RAID array device **100** illustrated in FIG. **1** may be comprised of four storage devices, but this is not intended to be limiting. Other examples may include more than four storage devices or may include fewer than four storage devices.
(5) Storage devices **101**, **102**, **103** and **104** may, respectively, be non-transitory storage devices, including but not limited to Dynamic Random Access Memory (DRAM), Non-Volatile Memory (NVM), Embedded Non-Volatile Memory (eNVM), Non-Volatile Memory Express (NVMe), or another type of non-transitory storage not specifically mentioned. Storage devices **101**, **102**, **103** and **104** may, respectively, include a DMA (Direct Memory Access) circuit to move data to and from the storage device.

(6) Storage devices **101**, **102**, **103** and **104** may, respectively, be coupled to memory bus **105**, which may be a Peripheral Component Interconnect Express (PCIe) bus. Memory bus **105** may be coupled to host processor **108**. Host processor **108** may be a central processing unit (CPU) or other processing device. Host processor **108** may drive signals onto memory bus **105** to read data from storage devices **101**, **102**, **103** and **104**, and may drive signals onto memory bus **105** to write data to storage devices **101**, **102**, **103** and **104**. Memory bus **105** may be a Peripheral Component Interconnect Express (PCIe) bus, or another bus type not specifically mentioned.
(7) Storage devices **101**, **102**, **103**, and **104** may be divided into strips. Each strip may be a predetermined number of bytes within the storage device. Strips within a storage device may be the same number of bytes, or may be different numbers of bytes. As one of various examples, storage device **101** may be divided into strips **111**, **121**, **131** and **141**, storage device **102** may be divided into strips **112**, **122**, **132** and **142**, storage device **103** may be divided into strips **113**, **123**, **133** and **143**, and storage device **104** may be divided into strips **114**, **124**, **134** and **144**.
(8) In operation, a block of data may be stored across strips in multiple storage devices. In the example illustrated in FIG. **1**, a block of data, called block A, may divided into portions A**1**, A**2** and A**3**. The first portion, A**1**, may be stored in storage device **101** at strip **111**, the second portion, A**2**, may be stored in storage device **102** at strip **112**, and the third portion, A**3**, may be stored in storage device **103** at strip **113**.
(9) At storage device **104**, strip **114** may store parity information based on the data stored in strips **111**, **112** and **113**, the parity information labelled Ap. In one of various examples, the parity information may be computed as a bitwise XOR of the data stored in strips **111**, **112** and **113**. In other examples, parity information may be computed using a different logical operation or an algorithmic method. Parity information may be computed by host processor **108**. In this manner, if any one of the storage devices **101**, **102**, **103** or **104** may be corrupted or damaged, the corrupted or damaged data may be recovered based on the data in the non-damaged storage devices.
(10) In the example illustrated in FIG. **1**, another block of data, block B, may be divided into portions B**1**, B**2** and B**3**. The first portion, B**1**, may be stored in storage device **101** at strip **121**, the second portion, B**2**, may be stored in storage device **102** at strip **122**, and the third portion, B**3**, may be stored in storage device **104** at strip **124**.
(11) At storage device **103**, strip **123** may store parity information based on the data stored in strips **121**, **122** and **124**, the parity information labelled Bp. In one of various examples, the parity information may be computed as a bitwise XOR of the data stored in strips **121**, **122** and **124**. In other examples, parity information may be computed using a different logical operation or an algorithmic method. Parity information may be computed by host processor **108**. In this manner, if any one of the storage devices **101**, **102**, **103** or **104** may be corrupted or damaged, the corrupted or damaged data may be recovered based on the data in the non-damaged storage devices.
(12) In the example illustrated in FIG. **1**, another block of data, block C, may be divided into portions C**1**, C**2** and C**3**. The first portion, C**1**, may be stored in storage device **101** at strip **131**, the second portion, C**2**, may be stored in storage device **103** at strip **133**, and the third portion, C**3**, may be stored in storage device **104** at strip **134**.
(13) At storage device **102**, strip **132** may store parity information based on the data stored in strips **131**, **133** and **134**, the parity information labelled Cp. In one of various examples, the parity information may be computed as a logical XOR of the data stored in strips **131**, **133** and **134**. In other examples, parity information may be computed using a different logical operation or an algorithmic method. Parity information may be computed by host processor **108**. In this manner, if any one of the storage devices **101**, **102**, **103** or **104** may be corrupted or damaged, the corrupted or damaged data may be recovered based on the data in the non-damaged storage devices.
(14) In the example illustrated in FIG. **1**, another block of data, block D, may be divided into portions D**1**, D**2** and D**3**. The first portion, D**1**, may be stored in storage device **102** at strip **142**, the second portion, D**2**, may be stored in storage device **103** at strip **143**, and the third portion, D**3**, may

be stored in storage device **104** at strip **144**, the data labelled D**3**.
(15) At storage device **101**, strip **141** may store parity information based on the data stored in strips **142**, **143** and **144**, the parity information labelled Dp. In one of various examples, the parity information may be computed as a logical XOR of the data stored in strips **142**, **143** and **144**. In other examples, parity information may be computed using a different logical operation or an algorithmic method. Parity information may be computed by host processor **108**. In this manner, if any one of the storage devices **101**, **102**, **103** or **104** may be corrupted or damaged, the corrupted or damaged data may be recovered based on the data in the non-damaged storage devices.
(16) In the example illustrated in FIG. **1**, the parity information is illustrated in specific strips in specific storage devices, but this is not intended to be limiting. Location of parity information may be stored in randomized strips, or may be stored in a deterministic arrangement.
(17) FIG. **2** illustrates one of various examples of a system **200** for computing parity information in a RAID array.
(18) CPU **210** may be coupled to a shared bus **250**. Shared bus **250** may receive data from CPU **210** and may transmit data to one or more circuits coupled to shared bus **250**. CPU **210** may be coupled to processor memory **220**. CPU **210** may write data to processor memory **220** and may read data from processor memory **220**. Shared bus **250** may be a PCIe bus, or another type of shared bus not specifically mentioned.
(19) Accelerator memory **230** may be coupled to shared bus **250**. Accelerator memory **230** may be a non-transitory memory. Accelerator memory **230** may be dedicated memory hardware, or may be a memory-mapped space allocated in a larger memory device. Shared bus **250** may write data to accelerator memory **230** and may read data from accelerator memory **230**. Accelerator processor **240** may receive data from shared bus **250** and may transmit data to shared bus **250**. Accelerator memory **230** and accelerator processor **240** may comprise an accelerator device.
(20) Storage device **281** may be a non-transitory storage device, including but not limited to DRAM, NVM, eNVM, NVMe, or another type of non-transitory storage not specifically mentioned. Storage device **281** may be coupled to shared bus **250**. Shared bus **250** may facilitate data transmission to and from storage device **281**. CPU **210** may move data through shared bus **250** to storage device **281**, and CPU **210** may receive data from storage device **281** through shared bus **250**. Storage device **281** may include DMA circuit **286**. DMA circuit **286** may control data read from storage device **281**. Particularly, data read from storage device **281** may be moved to a destination address provided with a read instruction received by storage device **281**. CPU **210** may move data through shared bus **250** to storage device **281**, and CPU **210** may receive data from storage device **281** through shared bus **250**. Storage device **281** may be partitioned into strips, as illustrated and described in reference to FIG. **1**.
(21) Storage device **281** may include DMA circuit **286**. CPU **210** may send a command to storage device **281**, the command to instruct DMA circuit **286** to read data from storage device **281** or to write data to storage device **281**. Based on the address sent in the command, data may be written to or read from processor memory **220** or accelerator memory **230**.
(22) Storage device **282** may be a non-transitory storage device, including but not limited to DRAM, NVM, eNVM, NVMe, or another type of non-transitory storage not specifically mentioned. Storage device **282** may be coupled to shared bus **250**. Shared bus **250** may facilitate data transmission to and from storage device **282**. CPU **210** may move data through shared bus **250** to storage device **282**, and CPU **210** may receive data from storage device **282** through shared bus **250**. Storage device **282** may be partitioned into strips, as illustrated and described in reference to FIG. **1**.
(23) Storage device **282** may include DMA circuit **287**. CPU **210** may send a command to storage device **282**, the command to instruct DMA circuit **287** to read data from storage device **282** or write data to storage device **282**. Based on the address sent in the command, data may be written to or read from processor memory **220** or accelerator memory **230**.

(24) Storage device **283** may be a non-transitory storage device, including but not limited to DRAM, NVM, eNVM, NVMe, or another type of non-transitory storage not specifically mentioned. Storage device **283** may be coupled to shared bus **250**. Shared bus **250** may facilitate data transmission to and from storage device **283**. CPU **210** may move data through shared bus **250** to storage device **283**, and CPU **210** may receive data from storage device **283** through shared bus **250**. Storage device **283** may be partitioned into strips, as illustrated and described in reference to FIG. **1**.

(25) Storage device **283** may include DMA circuit **288**. CPU **210** may send a command to storage device **283**, the command to instruct DMA circuit **288** to read data from storage device **283** or write data to storage device **283**. Based on the address sent in the command, data may be written to or read from processor memory **220** or accelerator memory **230**.

(26) Storage device **284** may be a non-transitory storage device, including but not limited to DRAM, NVM, eNVM, NVMe, or another type of non-transitory storage not specifically mentioned. Storage device **284** may be coupled to shared bus **250**. Shared bus **250** may facilitate data transmission to and from storage device **284**. CPU **210** may move data through shared bus **250** to storage device **284**, and CPU **210** may receive data from storage device **284** through shared bus **250**. Storage device **284** may be partitioned into strips, as illustrated and described in reference to FIG. **1**.

(27) Storage device **284** may include DMA circuit **289**. CPU **210** may send a command to storage device **284**, the command to instruct DMA circuit **289** to read data from storage device **284** or write data to storage device **284**. Based on the address sent in the command, data may be written to or read from processor memory **220** or accelerator memory **230**.

(28) Storage devices **281**, **282**, **283** and **284** may form a RAID array **280**. The example illustrated in FIG. **2** may form a RAID-5 array, but this is not intended to be limiting. Storage elements **281**, **282**, **283** and **284** may comprise a different type of storage array.

(29) In operation, system **200** may compute parity information in RAID array **280** and may rebuild damaged storage devices within RAID array **280**. Data may be read from one or more of storage devices **281**, **282**, **283** and **284**, and may be stored in a source buffer space in accelerator memory **230** based on an address provided in the read instruction issued by CPU **210** as part of a rebuild routine. The source buffer space may be a memory-mapped portion of a memory device, or may be a dedicated memory circuit. DMA circuit **286** may move data from storage device **281** to a predetermined source buffer location in accelerator memory **230**. DMA circuit **287** may move data from storage device **282** to a predetermined source buffer location in accelerator memory **230**. DMA circuit **288** may move data from storage devices **283** to a predetermined source buffer location in accelerator memory **230**. DMA circuit **289** may move data from storage devices **284** to a predetermined source buffer location in accelerator memory **230**, which predetermined source buffer location is defined in the read instruction provided by CPU **210**. In this manner, CPU **210** and processor memory **220** may store data from one or more storage devices **281**, **282**, **283** and **284** for the rebuild routine. CPU **210** resources may be reserved for other operations. Data read from the one or more storage devices **281**, **282**, **283** and **284** may comprise one or more strips of storage devices **281**, **282**, **283** and **284**.

(30) At the completion of the data move of the read data from one or more storage devices **281**, **282**, **283** and **284**, an interrupt may be issued to CPU **210** by the respective ones of one or more storage devices **281**, **282**, **283** and **284**. CPU **210** may issue a command to accelerator processor **240** over shared bus **250**, the command instructing accelerator processor **240** to perform a parity calculation on the data stored at the location in accelerator memory **230** where the read data moved from the one or more storage devices **281**, **282**, **283** and **284** has been stored. Accelerator processor **240** may access data in accelerator memory **230** directly, without requiring any commands or control from CPU **210**. After the parity information is calculated, the parity information may be stored in a results buffer in accelerator memory **230**. The results buffer may be a memory-mapped

space, or may be a dedicated memory circuit. At the completion of the parity calculation, the parity information stored in accelerator memory **230** may be written to one of storage devices **281**, **282**, **283** and **284** by one of DMA circuits **286**, **287**, **288** and **289**. In this manner, parity information may be computed for a RAID array without using CPU **210** to move data between storage devices **281**, **282**, **283**, **284** and accelerator memory **230**.
(31) FIG. **3** illustrates a method **300** for computing parity information in a RAID array, which may utilize system **200**.
(32) At operation **310**, a host processor may allocate source and result buffers in the host memory-mapped space belonging to the accelerator device. The host processor may be a CPU or other processor device. The buffers in the accelerator memory may be memory mapped input/output space. The host processor may be connected to the accelerator memory over a PCIe bus or another shared bus architecture not specifically mentioned.
(33) At operation **320**, the host processor may issue a command to read data from one or more storage devices to the source buffers. The movement of data may be performed by a DMA circuit in the respective storage devices.
(34) At operation **330**, the one or more storage devices may send an interrupt to the host processor at the completion of the data move. At operation **340**, the host processor may issue a command instructing the accelerator processor to compute parity information on the data stored in the source buffers.
(35) At operation **350**, the accelerator processor may perform the parity calculation on the data in the source buffers. In one of various examples the parity information is computed by an exclusive OR (XOR) bit by bit of the data moved to the source buffers. At operation **360**, the accelerator processor may write the parity result to the result buffer.
(36) At operation **370**, the host may issue a command to move the parity result from the result buffer to one or more storage devices.

Claims

1. A system comprising: a central processing unit (CPU); a non-transitory processor memory device coupled to the CPU; one or more storage devices, the one or more storage devices comprising a storage array and a Direct Memory Access (DMA) circuit; an accelerator processor; an accelerator non-transitory memory; a shared bus, shared bus coupled to the CPU, the accelerator processor, the accelerator non-transitory memory, and the one or more storage devices, and wherein the CPU to instruct the one or more storage devices to move data between the one or more storage devices and the accelerator non-transitory memory, and wherein the CPU to instruct the accelerator processor to perform a parity calculation on data stored in the accelerator non-transitory memory and to write a result of the parity calculation to the one or more storage devices.
2. The system as claimed in claim 1, the one or more storage devices partitioned into one or more strips, respective strips comprised of a predetermined number of bytes.
3. The system as claimed in claim 2, the strips comprised of portions of blocks of data.
4. The system as claimed in claim 2, the strips comprised of parity information.
5. The system as claimed in claim 1, the accelerator non-transitory memory comprising a source buffer and a result buffer.
6. The system as claimed in claim 5, the CPU to instruct the DMA circuit in one or more storage devices to move data between the one or more storage devices and the source buffer within the accelerator non-transitory memory.
7. The system as claimed in claim 5, the CPU to instruct the accelerator processor to write a result of the parity calculation to the result buffer within the accelerator non-transitory memory.
8. The system as claimed in claim 1, the system to rebuild the storage array based on data stored in the one or more storage devices.

9. A method comprising: allocating a source buffer and a result buffer in an accelerator memory in an accelerator device, the accelerator device external to one or more storage devices; moving data from the one or more storage devices to the source buffer; issuing an interrupt from the one or more storage devices to a host processor; calculating parity information based on data in the source buffer; writing the parity information to the result buffer in the accelerator memory, and writing the parity information to the one or more storage devices.
10. The method as claimed in claim 9, the one or more storage devices partitioned into one or more strips, each strip comprised of a predetermined number of bytes.
11. The method as claimed in claim 10, the strips comprised of portions of blocks of data.
12. The method as claimed in claim 10, the strips comprised of parity information.
13. The method as claimed in claim 9, the moving data from one or more storage devices to the source buffer to be performed by a respective DMA circuit of the one or more storage devices.
14. The method as claimed in claim 9, the writing the parity information to the one or more storage devices to be performed by a respective DMA circuit of the one or more storage devices.
15. The method as claimed in claim 9, the calculating parity information based on data in the source buffer to be performed by an accelerator processor.