

Description

SAFE WRITE TO MULTIPLY-REDUNDANT STORAGE

Technical Field

- [001] The present invention relates to the field of storage, and more specifically to the operation of multiply-redundant arrays.

Background Art

- [002] In storage systems an array of independent storage devices can be configured to operate as a single virtual storage device using a technology known as RAID (Redundant Array of Independent Disks). A computer system configured to operate with a RAID storage system is able to perform input and output (I/O) operations (such as read and write operations) on the RAID storage system as if the RAID storage system were a single storage device. A RAID storage system includes an array of independent storage devices and a RAID controller. The RAID controller provides a virtualized view of the array of independent storage devices - this means that the array of independent storage devices appear as a single virtual storage device with a sequential list of storage elements. The storage elements are commonly known as blocks of storage, and the data stored within them are known as data blocks. I/O operations are qualified with reference to one or more blocks of storage in the virtual storage device. When an I/O operation is performed on the virtual storage device the RAID controller maps the I/O operation onto the array of independent storage devices. In order to virtualize the array of storage devices and map I/O operations the RAID controller may employ any of several standard RAID techniques well known to those skilled in the data processing art.
- [003] In providing a virtualized view of an array of storage devices as a single virtual storage device it is a function of a RAID controller to spread data blocks in the virtual storage device across the array. One way to achieve this is using a technique known as Striping. Striping involves spreading data blocks across storage devices in a round-robin fashion. When storing data blocks in a RAID storage system, a number of data blocks known as a strip is stored in each storage device. The size of a strip may be determined by a particular RAID implementation or may be configurable. A row of strips comprising a first strip stored on a first storage device and subsequent strips stored on subsequent storage devices is known as a stripe. The size of a stripe is the total size of all strips comprising the stripe. The use of multiple independent storage devices to store data blocks in this way provides for high performance I/O operations when compared to a single storage device because multiple storage devices can act in parallel during I/O operations.

[004] Physical storage devices such as disk storage devices are renowned for poor reliability and it is a further function of a RAID controller to provide a reliable storage system. One technique to provide reliability involves the storage of check information along with data in an array of independent storage devices. Check information is redundant information that allows regeneration of data which has become unreadable due to a single point of failure, such as the failure of a single storage device in an array of such devices. Unreadable data is regenerated from a combination of readable data and redundant check information. Check information is recorded as parity data which occupies a single strip in a stripe, and is calculated by applying the EXCLUSIVE OR (XOR) logical operator to all data strips in the stripe. For example, a stripe comprising data strips A, B and C would be further complimented by a parity strip calculated as $A \text{ XOR } B \text{ XOR } C$. In the event of a single point of failure in the storage system, the parity strip is used to regenerate an inaccessible data strip. For example, if a stripe comprising data strips A, B, C and PARITY is stored across four independent storage devices W, X, Y and Z respectively, and storage device X fails, strip B stored on device X would be inaccessible. Strip B can be computed from the remaining data strips and the PARITY strip through an XOR computation. This restorative computation is $A \text{ XOR } C \text{ XOR } \text{PARITY} = B$.

[005] A multiply redundant RAID array is a storage device built out of independent hard disk drives which stores data and parity information in such a way that several of the component hard disk drives can fail without losing any of the data written to the array. For simplicity, this will be described here in terms of doubly redundant arrays; the techniques of the preferred embodiments of the present invention can be extended to arrays with more redundancy.

[006] For example, there is shown in Figure 1 a well known scheme to support the restorative XOR technique described above for a four-disk array. The technique relies on the following:

- [007] 1. Each column represents a disk.
- [008] 2. Each cell in the figure represents a strip of data (top row) or parity (bottom row) on the disk.
- [009] 3. The pattern repeats to cover the entire array.
- [010] 4. Each data strip contributes to exactly two parity strips.
- [011] 5. Each data strip and its two parity strips lie on different disks.
- [012] 6. No two data strips share the same two parity strips.
- [013] 7. If any two disks (columns) are removed, the missing data can be recovered from the remaining two disks.

[014] For example, remove the first (left-hand) two disks. It is possible to recover A from:

[015] $C \text{ xor } (A \text{ xor } C)$.

[016] It is then also possible to recover B from:

[017] $A \text{ xor } (A \text{ xor } B)$.

[018] When a write operation is interrupted (e.g. by a power failure or reset of the array controller,) the array controller must take steps to resynchronize the pattern. For example, consider overwriting data B above with data B'. This means that three strips (one data and two parity) have to be written. If power is lost during these writes, the disk drives will not complete the writes and the strips will be left containing a mixture of old and new data or parity. When power is returned, the array controller must take some action to resynchronize the parity strips to the data strips -- if this is not done then the pattern will not reconstruct correctly and data corruption will occur.

[019] Current RAID adapters which implement RAID-5, a singly redundant type of RAID, typically manage this by storing in a small non-volatile memory (NVRAM) the indices of each stripe being written. When the adapter starts up, it looks through its NVRAM and discovers which of the patterns on the array need to be resynchronized. This resynchronization is done by reading all of the data strips and computing the new parity. This approach needs ~32 bits of NVRAM for each pattern that needs to be written concurrently. This technique only works if all of the data strips can be read. If a single data strip cannot be read (eg. because a disk has failed,) the parity strip cannot be computed and data is lost. This borders on the acceptable with RAID-5, which is designed to survive a single failure (there have arguably been two failures, the disk failure and the power failure or controller reset.)

[020] This technique is not acceptable when applied to doubly-redundant arrays, which are supposed to survive two failures without losing data. The normal solution to this problem (which enhances the RAID-5 case and solves the doubly-redundant case) is to equip the array controller with a large amount of non-volatile storage (referred to here as NVS to distinguish it from NVRAM.) The controller prepares the new data and parity and stores them in NVS before starting to write to the disks. If the controller is reset or the disks lose power during the writes, the controller is able to try the writes again because the data it was attempting to write is still present in NVS. If the writes succeed, no further action is required to resynchronize the pattern. Even if one disk (or even two disks for doubly-redundant arrays) fails, there is no problem.

[021] The drawback to this approach is that it requires a large amount of NVS -- enough to cope with as many concurrent writes as the controller needs to perform acceptably. In the past this has been acceptable because a single storage controller typically supplied both fastwrite cache and RAID function -- only one piece of NVS, shared by the two components, is required.

[022] In a storage area network (SAN) environment, however, it may be desirable to have cache and RAID in separate boxes. Using NVS for RAID updates means that both the

cache and RAID boxes need their own NVS.

[023] Typically there are at least two controllers of a RAID array, so that failure of a single controller does not cause loss of access to the array. These controllers must communicate to synchronise access to the RAID array. Any non-volatile information must be visible to both controllers. In the simplest case, the controllers communicate using the device network, but if the contents of the NVS are to be shadowed, network bandwidth is disadvantageously consumed, which bandwidth would otherwise be available for user I/O.

[024] It is a further disadvantage that given a storage adapter/controller which is managing RAID-5 arrays using the NVRAM technique, it is not possible to implement acceptable doubly redundant RAID storage using the NVS approach without a hardware upgrade.

Disclosure of Invention

[025] The present invention accordingly provides, in a first aspect an arrangement of apparatus for safely writing data and parity to multiply-redundant storage comprising a first storage component operable to store at least a first mark in a storage device to index uniquely a pattern to be written by at least a data write; a write component operable to perform said at least data write; a further storage component operable to overwrite a mark in said storage device with at least a further mark to index uniquely a pattern to be written by a parity write; and a further write component operable to perform said parity write.

[026] Preferably, said first storage component further comprises a second storage component operable to overwrite said at least first mark in said storage device with a second mark to index a pattern to be written by a first parity write; and said write component is further operable to perform said first parity write.

[027] The arrangement of the first aspect may be preferred to have double redundancy. The apparatus of the first aspect may be preferred to have

[028] greater than double redundancy.

[029] The arrangement of the first aspect may be adapted to serialise each data and parity write operation.

[030] The arrangement of the first aspect may further comprise RAID storage, which may preferably be RAID-5 storage.

[031] The arrangement of the first aspect may be adapted to perform lazy parity update.

[032] The arrangement of the first aspect may preferably further comprise a fastwrite cache.

[033] In a second aspect, the present invention provides a method for safely writing data and parity to multiply-redundant storage comprising storing at least a first mark in a

storage device to index uniquely a pattern to be written by at least a data write; performing said at least data write; overwriting a mark in said storage device with at least a further mark to index uniquely a pattern to be written by a parity write; and performing said parity write.

[034] Preferably, said step of storing at least a first mark in a storage device to index uniquely a pattern to be written by at least a data write further comprises overwriting said at least first mark in said storage device with a second mark to index a pattern to be written by a first parity write; and said step of performing said at least data write comprises performing said first parity write.

[035] Preferably, the multiply redundant storage is doubly redundant.

[036] Preferably, the multiply redundant storage has greater than double redundancy.

[037] Preferably, all steps comprising performing a write are serialised.

[038] Preferably, said multiply redundant storage comprises RAID storage.

[039] Preferably, said RAID storage is RAID-5 storage.

[040] The method of the second aspect preferably further comprises a step of performing a lazy parity update.

[041] The method of the second aspect preferably further comprises a step of performing a fastwrite caching operation.

[042] In a third aspect, the present invention provides a computer program comprising computer program code embodied in a tangible medium, to, when loaded into a computer system and executed thereon, perform all the steps of the method of the second aspect.

[043] Advantageously, the present invention provides protection against a combination of a disk failure and a power reset or of a disk failure and a controller reset.

[044] Further advantageously, the present invention alleviates the problem of bandwidth consumption reducing the available bandwidth for user I/O.

[045] A further advantage of the present invention is that it is scalable from doubly-redundant arrays to arrays having higher orders of redundancy.

Brief Description of the Drawings

[046] A preferred embodiment of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

[047] Figure 1 shows a well known encoding scheme for a four-disk array;

[048] Figure 2 shows a block schematic of an arrangement of apparatus according to a preferred embodiment of the present invention.

[049] Figure 3 shows the steps of a method according to a preferred embodiment of the present invention.

Mode for the Invention

- [050] To appreciate the preferred embodiments of the present invention, consider again the known encoding scheme shown in Figure 1. Writes to this pattern are serialised. If writes for both strips A and B are submitted at the same time, the array controller will do one first and then the other.
- [051] To write updated data B' over the existing data B, there is a known sequence of disk I/O steps required. This sequence is:
- [052] 1. Read the old data, B.
- [053] 2. Calculate the parity delta, $B \text{ xor } B'$.
- [054] 3. Read the first old parity, $B \text{ xor } D$.
- [055] 4. Calculate the first new parity from the first old parity xor'd with the parity delta -- $(B \text{ xor } D) \text{ xor } (B \text{ xor } B')$ which is $B' \text{ xor } D$.
- [056] 5. Read the second old parity, $A \text{ xor } B$.
- [057] 6. Calculate the second new parity from the second old parity xor'd with the parity delta -- $(A \text{ xor } B) \text{ xor } (B \text{ xor } B')$ which is $A \text{ xor } B'$.
- [058] 7. Write B' on top of B. <----- Write #0
- [059] 8. Write the first new parity, $B' \text{ xor } D$. <----- Write #1
- [060] 9. Write the second new parity, $A \text{ xor } B'$. <----- Write #2
- [061] (It will be clear to one skilled in the art that some reordering is possible without affecting the end result.)
- [062] In the preferred embodiments of the present invention, a mark is stored in NVRAM to uniquely identify the strip within the pattern that is being written on each of writes #0, #1, and #2.
- [063] Turning to Figure 2, there is shown an arrangement of apparatus 100 for safely writing data and parity to multiply-redundant storage comprising a first write component 112 comprising a writer 102 which cooperates with a first storage component 108 to store a mark to index uniquely a pattern that is to be written by a data write, and which then performs a write operation to storage (110). Write component 112 may further contain a second writer 104 which also cooperates with a first storage component 108 to store a mark to index uniquely a pattern that is to be written by a first parity write, and which then performs a write operation to storage 110. In the case of a non-atomic write to storage, the data and parity may be written in parallel as shown by parallel writes 114. In the case of a write which must preserve atomicity, component 112 is effectively decomposed to provide two independent writers 102, 104, and the writes shown in the parallel write component 112 are decomposed into a serialized pair of writes, the first of data, and the second of the first parity (and note that the second parity write must also be serialized).
- [064] A third writer 106 cooperates with storage component 108 to overwrite the mark in storage with at least a further mark to index uniquely a pattern to be written by a

second parity write, and then writes the parity to storage 110.

[065] The information stored in storage component 108, which is an NVRAM is:

[066] The index of the pattern being written (0 through the number of times the pattern is repeated down the array);

[067] The index of the data strip being written (0 through 3 with a 4-disk array); and

[068] The index of the write (marked #0 to #2 above for a doubly redundant array) currently being executed.

[069] The second and third items uniquely identify which strip within the pattern is being written. There are alternative ways of representing this information, including, for example but not limited to, the use of X-Y coordinates within the pattern. It will be clear to one skilled in the data processing art that there are many alternative representations.

[070] This means that the data stored in NVRAM is changed before each write. The use of the NVRAM mark described above involves adding the following steps to the algorithm above:

[071] 6A. Set NVRAM to {Pattern P, Data B, Write 0}

[072] 7. Write B' on top of B

[073] 7A. Set NVRAM to {pattern P, Data B, Write 1}

[074] 8. Write the first new parity, B' xor D.

[075] 8A. Set NVRAM to {Pattern P, Data B, Write 2}

[076] 9. Write the second new parity, A xor B'.

[077] 9A. Erase NVRAM mark.

[078] In an environment where there is more than one adapter/controller sharing the array, the value of the NVRAM marks must be changed on all of the adapters/controllers.

[079] Turning to Figure 3, there is shown a flow diagram of the steps of a method according to the preferred embodiments of the present invention.

[080] At step 202, a mark is written to NVRAM to store the pattern index, strip index and write index for the data write. The data is written at step 204. At step 206, a mark is written to NVRAM to store the pattern index, strip index and write index for the first parity write. The first parity is written at step 208. In the case of a non-atomic update, these writes may occur in parallel. Otherwise, they must be serialized, and the second parity write (to be discussed below) must also be serialized.

[081] At step 210, a mark is written to NVRAM to store the pattern index, strip index and write index for the second parity write. The second parity is written at step 212. At step 214, the NVRAM is invalidated.

[082] If the subsystem loses power or the adapter/controller is reset at any point, without a disk failure, the pattern can easily be resynchronized.

[083] If there is a disk failure, and that disk contains the data or parity strip being written

at the time of the reset, the pattern can be resynchronized because all strips on the remaining disks are already in sync.

[084] If there is a disk failure and that disk is different from the disk containing the data or parity strip being written at the time of the reset, the pattern can be resynchronized as follows:

[085] 1. If the NVRAM says "Write 0" then the situation is similar to one where two disks have failed -- the one that has really failed and the one containing the data strip being written. Because the pattern is multiply redundant, all data can be reconstructed, returning the pattern to the state it was in before the write started.

[086] 2. If the NVRAM says "Write 2" then the situation is similar to one where two disks have failed -- the one that has really failed and the one containing the second parity strip. Because the pattern is multiply redundant, all data can be reconstructed, completing the write that was interrupted.

[087] 3. If the NVRAM says "Write 1" then there are three cases:

[088] A) The failed disk contains the data strip that was written during "Write 0".

[089] This is the same as a two-disk failure -- the disk that has really failed and the one containing the "Write 1" data.

[090] The redundancy of the pattern guarantees that all data can be reconstructed, returning the pattern to the state it was in before the write started.

[091] B) The failed disk contains the second parity strip that was to be written during "Write 2".

[092] This is the same as a two-disk failure -- the disk that has really failed and the one containing the "Write 1" data.

[093] The redundancy of the pattern guarantees that all data can be reconstructed, completing the interrupted write.

[094] C) The failed disk is a different disk.

[095] The data strips on the failed disk can always be reconstructed and the pattern resynchronized to complete the interrupted write.

[096] Each data strip on the failed disk has two parity strips. Neither of these is on the failed disk.

[097] At most one of these parity strips is shared with the data strip written during "Write 0", and so there is always at least one way of reconstructing the data strip.

[098] The first preferred embodiment of this invention requires that although the various reads may be issued in parallel, the three writes must be issued sequentially -- write #2 is not begun until write #1 has completed etc.

[099] One disadvantage of this embodiment is that the three disk writes are serialised, meaning that a write sent to the array will take longer than a write to the equivalent array using the NVS approach, where the three disk writes may be issued in parallel.

[100] This may be alleviated somewhat if the RAID array is behind a fastwrite cache and so at least partially isolated from host response time.

[101] An additional potential method of alleviating this disadvantage is by signalling successful completion to the host write I/O after step 8A -- this is the equivalent of the well-known "Lazy parity update" technique often applied to RAID-5.

[102] In a second embodiment of the present invention, the serialisation is unnecessary unless the application requires the additional property that a write to the RAID array be atomic. If it is acceptable for an interrupted write to end with a transition between new data and old data then the first two writes can occur under the first NVRAM mark and then one write can occur under a second NVRAM mark.

[103] For example consider the steps of an update to data A of Figure 1:

[104] 1. Do all the reads and calculate the new parity to be written

[105] 2. Establish NVRAM mark indicating A and A xor B are being updated

[106] 3. Write A and A xor B in parallel

[107] 4. When both writes complete establish NVRAM mark indicating A xor C is being updated

[108] 5. Write A xor C

[109] 6. Clear NVRAM mark

[110] So if a controller reset and a disk failure were to occur during the parallel write step number 3 the scenarios are:

[111] 1. Disk 1 is lost. "B", "C" and "D" remain and "C" and "A xor C" can be used to reconstruct the old version of "A".

[112] 2. Disk 2 is lost. "C", "D" and a half updated version of "A" remain "D" and "B xor D" can be used to reconstruct "B". "C" and

[113] "A xor C" can be used to reconstruct the old version of "A" if required.

[114] 3. Disk 3 is lost. "B", "D" and a half updated version of "A" remain. "D" and "C xor D" can be used to reconstruct "C". "C" and

[115] "A xor C" can be used to reconstruct the old version of "A" if required.

[116] 4. Disk 4 is lost. "B", "C" and a half updated version of "A" remain. "B" and "B xor D" can be used to reconstruct "D". We cannot reconstruct the old or new version of "A".

[117] This reduction in serialisation is desirable because it lowers the latency on the array write operation to that approaching current RAID-5 arrays.

[118] To extend the second preferred embodiment to arrays with more redundancy, the data write can be done in parallel with the first parity update, but the subsequent parity updates must be serialised.

[119] However, making the write atomic -- as in the first preferred embodiment involving three sequential writes -- would be desirable for USE with disk drives that do not

guarantee that an interrupted write can be read (e.g., a drive with a 4 KB sector size that allows sub-sector writes without using non-volatile storage).

[120] It will be appreciated that the method described above will typically be carried out in software running on one or more processors (not shown), and that the software may be provided as a computer program element carried on any suitable data carrier (also not shown) such as a magnetic or optical computer disc. The channels for the transmission of data likewise may include storage media of all descriptions as well as signal carrying media, such as wired or wireless signal media.

[121] The present invention may suitably be embodied as a computer program product for use with a computer system. Such an implementation may comprise a series of computer readable instructions either fixed on a tangible medium, such as a computer readable medium, for example, diskette, CD-ROM, ROM, or hard disk, or transmittable to a computer system, via a modem or other interface device, over either a tangible medium, including but not limited to optical or analogue communications lines, or intangibly using wireless techniques, including but not limited to microwave, infrared or other transmission techniques. The series of computer readable instructions embodies all or part of the functionality previously described herein.

[122] Those skilled in the art will appreciate that such computer readable instructions can be written in a number of programming languages for use with many computer architectures or operating systems. Further, such instructions may be stored using any memory technology, present or future, including but not limited to, semiconductor, magnetic, or optical, or transmitted using any communications technology, present or future, including but not limited to optical, infrared, or microwave. It is contemplated that such a computer program product may be distributed as a removable medium with accompanying printed or electronic documentation, for example, shrink-wrapped software, pre-loaded with a computer system, for example, on a system ROM or fixed disk, or distributed from a server or electronic bulletin board over a network, for example, the Internet or World Wide Web.

[123] It will be appreciated that various modifications to the embodiment described above will be apparent to a person of ordinary skill in the art.

Claims

- [001] An arrangement of apparatus for writing data and parity to multiply-redundant storage comprising: a first storage component operable to store at least a first mark in a storage device to index uniquely a pattern to be written by at least a data write; a write component operable to perform said at least data write; a further storage component operable to overwrite a mark in said storage device with at least a further mark to index uniquely a pattern to be written by a parity write; and a further write component operable to perform said parity write.
- [002] An arrangement of apparatus as claimed in claim 1, wherein said first storage component further comprises: a second storage component operable to overwrite said at least first mark in said storage device with a second mark to index a pattern to be written by a first parity write; and said write component is further operable to perform said first parity write.
- [003] An arrangement of apparatus as claimed in claim 1 or claim 2, having greater than double redundancy.
- [004] An arrangement of apparatus as claimed in claim 2, adapted to serialise each data and parity write operation.
- [005] A method for safely writing data and parity to multiply-redundant storage comprising: storing at least a first mark in a storage device to index uniquely a pattern to be written by at least a data write; performing said at least data write; overwriting a mark in said storage device with at least a further mark to index uniquely a pattern to be written by a parity write; and performing said parity write.
- [006] The method as claimed in claim 5, wherein said step of storing at least a first mark in a storage device to index uniquely a pattern to be written by at least a data write further comprises: overwriting said at least first mark in said storage device with a second mark to index a pattern to be written by a first parity write; and said step of performing said at least data write comprises performing said first parity write.
- [007] The method as claimed in claim 10 or claim 11, wherein the multiply redundant storage has greater than double redundancy.
- [008] The method as claimed in any one of claims 10 to 13, wherein all steps comprising performing a write are serialised.
- [009] The method as claimed in any one of claims 10 to 14 wherein said multiply redundant storage comprises RAID storage.
- [010] A computer program comprising computer program code embodied in a tangible medium, to, when loaded into a computer system and executed thereon, perform

all the steps of the method as claimed in any one of claims 5 to 9.