W H I T E P A P E R

Shivendra Singh,
Applications Engr. Principal
Cypress Semiconductor Corp.

nvSRAM as Write Journal in RAID Storage Systems

Abstract

The white paper explains the function of a RAID Storage System write journal and shows why nvSRAM is the most commonly chosen nonvolatile random access memory for this application.

RAID Storage Systems Overview

RAID was first described by Garth Gibson, Randy Katz, and Dave Patterson of UC Berkeley in the late 1980s as a 'Redundant Array of Inexpensive Disks'. It has now become a blanket term for data storage schemes that implement redundancy and parallelism for better fault tolerance and input/output (I/O) performance than a single disk drive or 'Just a Bunch Of Disks' (JBOD). RAID commonly represents a system consisting of many storage disks that are accessed by servers through high-speed Ethernet or Fibre Channel (FC) media. Disk drive I/O speed is the primary performance bottleneck in most storage systems. Generally, RAID Systems use parallel disk access and cache memories to increase the read and write performance, while using redundant disks for failure recovery to enhance fault tolerance.

Cache memories (especially write caches) in a RAID system are often backed up using a battery to avoid loss of data due to power failure. In many such systems, a write journal is used to keep track of data while it is being transferred from the host to the storage system and allows rapid system recovery in case power or disk failure occurs before the data is committed to the disks. If power is lost while write transactions are in progress, the journal memory is read back on power up to identify the pending writes. A nonvolatile memory with high speed and very high endurance is needed for the write journaling function since the journal is constantly accessed by the RAID controller for writing logs. Further, reliability of the data stored in this memory is critical, as losing this data can lead to long downtimes of the storage system in case of power failures.

nvSRAM

The nvSRAM (Nonvolatile Static Random Access Memory) is a fast nonvolatile memory with an SRAM interface. All reads and writes to the nvSRAM are done to the SRAM which gives it the unique ability to perform infinite read and write operations at very high speeds. If the power goes down, the data held in SRAM is transferred into the nonvolatile (NV) elements integrated with each SRAM cell. On power up, the data is automatically transferred back to the SRAM. High speed, infinite write/read endurance, and high reliability make nvSRAM an ideal nonvolatile memory solution for write journaling applications.

Storage Architectures

There are three storage architectures – DAS, SAN, and NAS – that are commonly used in the industry. DAS or Direct Attached Storage is used in personal computers or low-end servers. SAN and NAS are network-based storage architectures where servers use a shared set of storage drives through a network. There are also storage products that integrate the SAN and NAS functionality and are referred to as Unified Storage Solutions. This section details these architectures for data storage.

Direct Attached Storage (DAS)

This is the simplest form of data storage where the storage drives are connected directly to the server and each server has its own dedicated storage drives. The server accesses the storage drives through a direct physical interface such as serial attached ATA (SATA), small computer system interconnect (SCSI), or serial attached SCSI (SAS). In this scheme, the storage area is not optimally used and there is no possible load balancing. Further, as the memory needs of the servers increase, each server's storage requirements must be individually managed. This creates significant overhead in managing DAS for large server systems. However, this scheme is still commonly used in personal computers and small-scale server systems.

Figure 1. Disk Attached Storage

Storage Area Network (SAN)

Storage area networks consist of a shared storage system that is interfaced with servers through an Ethernet or FC network. SAN solves the problems related to load balancing of storage capacity and eliminates the overhead involved in managing different storage blocks. However, the biggest challenge for SAN systems is to ensure that disk I/O performance is not degraded due to access through the network.

When SAN systems were first introduced, FC networks gained popularity as Ethernet did not offer the speed required to make SAN effective. However, dedicated FC networks had to be laid out for SAN systems, making them very expensive. With increasing Ethernet speeds, Ethernet/Internet based systems have become very popular. Ethernet-based protocols like internet SCSI (iSCSI) or FC over IP (FCIP), lower the cost of implementing a SAN making it possible for even smaller firms to use these systems.

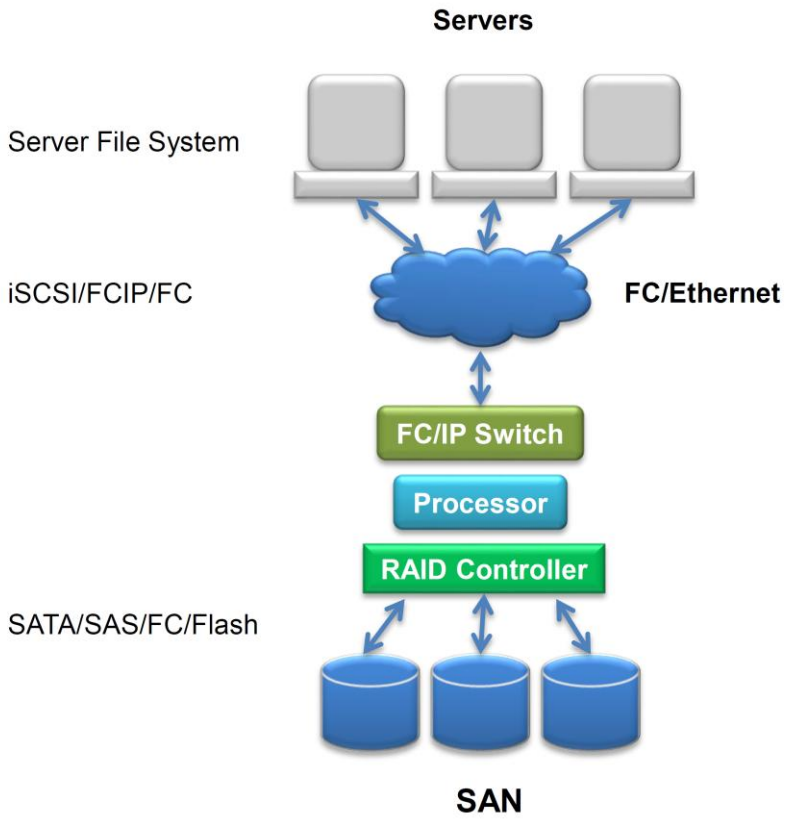

Figure 2. Storage Area Network

Data transfers from servers to the storage system are done using block I/Os, which are the same as the physical layer protocols used in disks like SATA, SCSI, SAS, and FC. This enables the servers in a SAN to use their own file systems. Their interaction with the storage drives is through a virtualization scheme, which makes the interaction appear to be on a physical layer as in a DAS.

The cost of maintaining such systems is low. It enables different servers using different Operating Systems or file systems to be connected to the same storage system. There is also no additional overhead on the server side and multiple servers can be added with a low incremental cost. At the same time, the virtualization of storage area enables the addition of capacity to be added without affecting the functioning of the servers or the availability of the storage area.

Network Attached Storage (NAS)

Network attached storage systems were first introduced when SAN was still implemented on FC networks. NAS used Ethernet/Internet for interacting with storage systems, which was the main advantage of this scheme. However, with the definition of SAN expanding to include Ethernet based protocols like iSCSI, the difference between SAN and NAS is converging.

A NAS system is attached to the network and the servers can access the storage through file I/O over Ethernet. The NAS system handles the physical access to the disks and presents the storage system to the servers at the file system level. This system, therefore, requires the servers to implement dedicated file systems like Network File System (NFS) or common internet file system (CIFS). The principal disadvantage of using NAS is that any new server being added to the network must support the file system specified by NAS. However, NAS is popularly used in applications where file I/O is the preferred mode of communication by the servers.

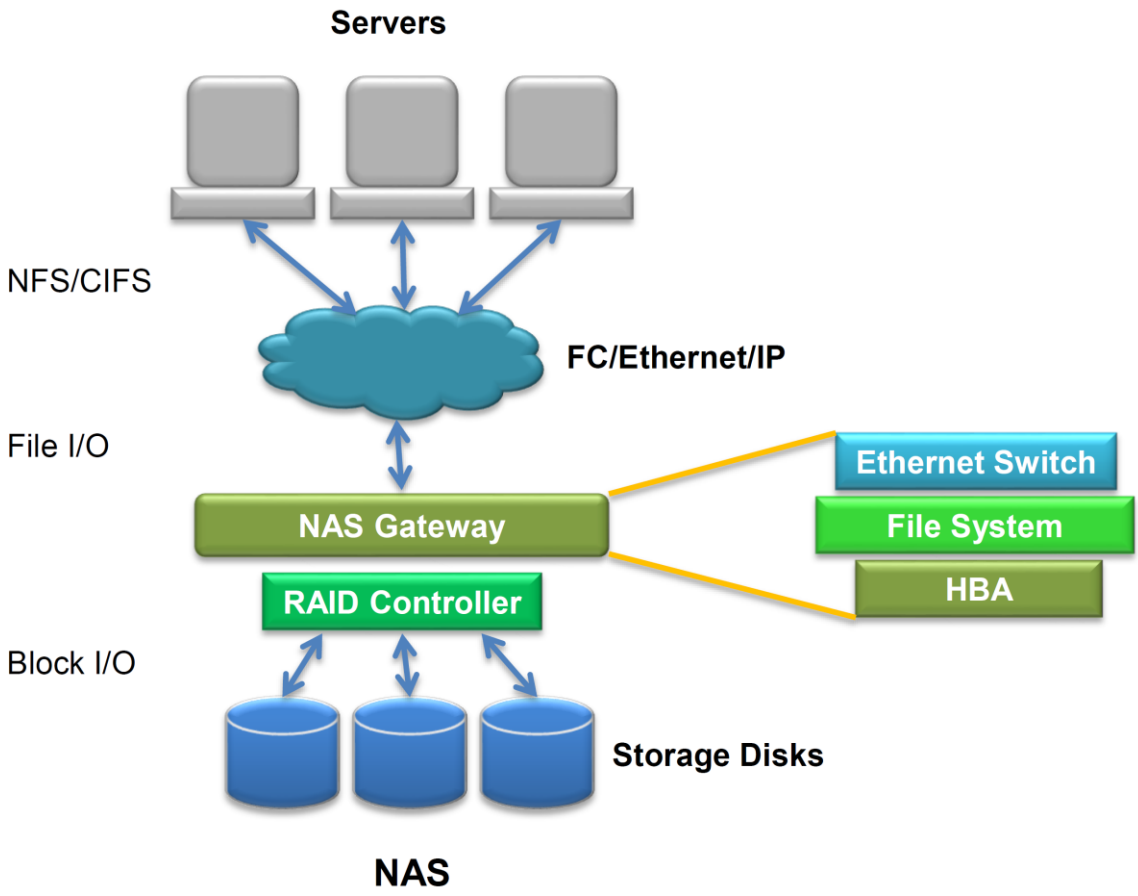

Figure 3. Network Attached Storage Systems

RAID (Redundant Array of Inexpensive/Independent Disks)

When RAID was first introduced by the Berkeley team in late 1980s, the term was defined as "Redundant Array of Inexpensive Disks". The I/O performance of disk drives has always been the major bottleneck for any system requiring access to huge amounts of storage and there were several efforts being made at that time to increase the throughput of disks through mechanical means. Such efforts, however, led to cost increases of these high-density, high-speed disks making it unaffordable for small businesses. One solution was to use many lower density, low cost drives operating in parallel, but this created a new problem of data reliability and recovery from disk failures.

RAID was the most popularly accepted solution to the problem as it was based purely on principles of computer science – redundancy and parallelism – and allowed the use of inexpensive storage media to achieve very high performance and safety from disk failures. The popularity of RAID was further boosted by the success of the Internet which generated huge demand for high performance data storage and management systems. Over time, the definition of RAID has changed to "Redundant Array of Independent Disks" making the term more generic and applicable to higher cost disk drives like SAS, solid state drives, and others.

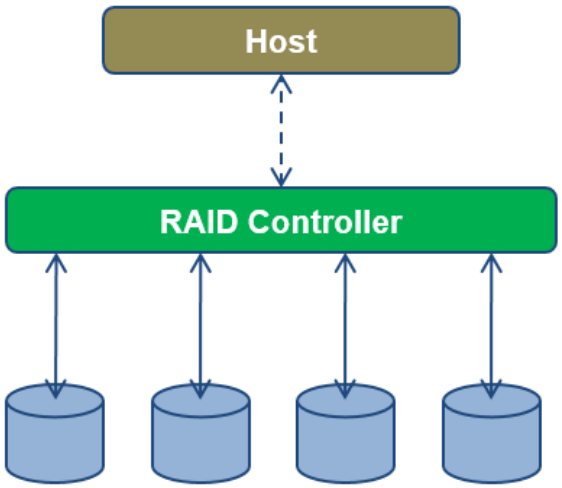

Figure 4. RAID Architecture

There are different implementations of RAID systems suitable for different applications Most of these implementations were suggested in the original Berkeley RAID paper and some were invented based on improvisations and industry needs. These different implementations are referred to as RAID 'Levels'. All the RAID levels, are however, based on a few basic principles like parallelism, redundancy, and duplication. The following section details the principles that enable RAID.

Data Striping

The concept of striping is used to split a large data block into smaller chunks to be stored on multiple disk drives. This is the most important concept used for increasing the disk performance through parallel I/O of data to disks. Reading and writing from multiple disks reduces the overall latency of disks and enables faster access to data.

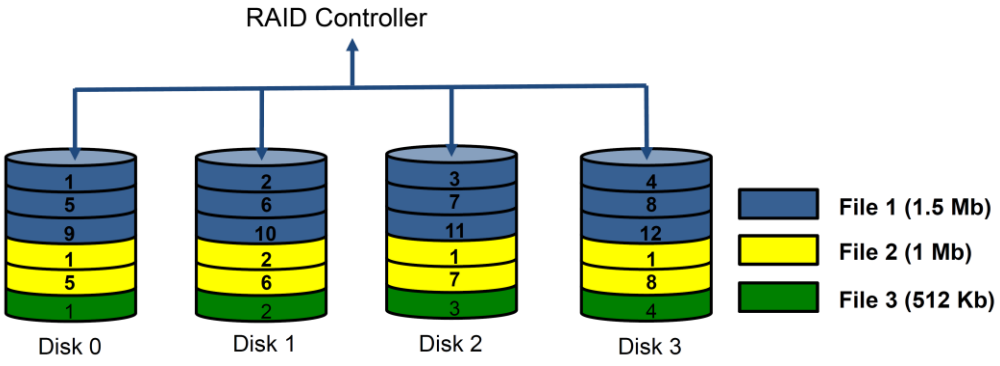

Figure 5. Data Striping

Figure 5 demonstrates the concept of striping in a system with four physical disks. The files (or data blocks) are split across these disks in a constant size of 128 Kb. The stripe depth (or the size of a stripe) in this case is 128 Kb. The figure shows how the files are striped across the drives. The stripe width (or the number of drives over which striping is done) dictates the performance improvement due to striping. The wider the stripe, the higher the I/O performance of the system.

Pros: High Read and Write Performance

Cons: Computational overhead for data striping, no fault tolerance

Data Mirroring

Mirroring is the simplest of RAID redundancy techniques where each write is done to two separate disks. A mirrored system greatly reduces the chance of failure of the whole system. If a disk fails, the data can be recovered from the mirrored disk enabling easy recovery. However, this causes very high overhead on memory usage and therefore is not commonly used. This technique is used by applications requiring very high fault tolerance and can afford the memory usage overhead.

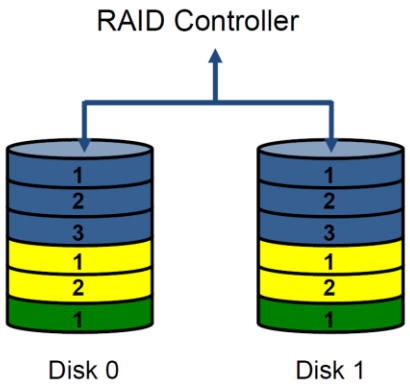

Figure 6. Data Mirroring

Pros: Very High Fault Tolerance; slightly higher read performance as reads can be spread across two drives.

Cons: 50 percent memory efficiency; Writes are a little slower due to computational overhead for mirroring.

Parity (Data Encoding)

A parity based RAID scheme builds redundancy by generating a parity block using data from each block in a stripe. This parity block is written along with the data blocks. This enables recovery from failure of one or more disks depending on the type of parity used. A single parity scheme has an overhead of one extra disk drive and enables recovery from one disk failure. Double parity schemes can recover from two simultaneous disk failures and have an overhead of two disk drives per array.

Parity based redundancy schemes are commonly used in the industry due to their low overhead and ability to recover from most commonly occurring failure scenarios. The possibility of simultaneous disk failures is extremely low which makes parity based schemes very attractive. Even the most commonly used RAID implementation in the industry (RAID Level 5) uses single bit parity for fault tolerance. Systems requiring higher fault tolerance may use double or multiple parity schemes.

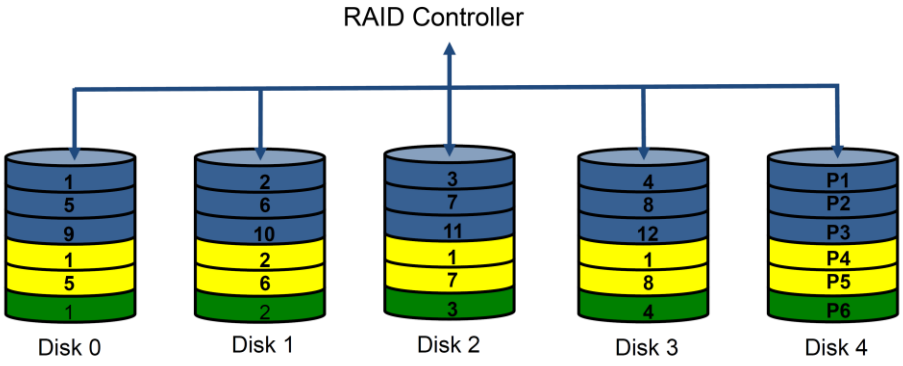

Figure 7. Single Parity Scheme

Pros: High Fault Tolerance, High data read performance

Cons: Slower writes due to computational overhead for parity calculation.

Other RAID Techniques

Duplexing: Duplexing is a more fault tolerant version of mirroring, where the storage controller and cache are also mirrored to avoid system failures due to storage controller failure.

Reed-Solomon: This is an ECC technique used for data encoding in double or multiple parity systems. This provides higher fault tolerance than single parity schemes, but increases the processing power required for calculating the parity.

RAID Levels

There are multiple implementations of RAID in the industry today. However, all the popular RAID implementations use one or more of the techniques described above. The different implementations of RAID systems are referred to as RAID levels. Although it may be intuitive to do so, the RAID level number must not be confused as effectiveness of a particular RAID technique. Each RAID level has its merits and demerits and must be chosen on its merits for a particular application. The most popular RAID levels being used in industry today are discussed below:

RAID Level 0

This is also referred to as AID because it does not have any redundancy built in. RAID Level 0 is a high performance RAID system with zero fault tolerance. It incorporates data striping to increase the read and write performance of the system. However, no redundancy means the memory efficiency is 100 percent.

Pros: Very high performance; 100 percent memory efficiency

Cons: No Fault Tolerance; Processing overhead for striping of data

RAID Level 1

This RAID level uses data mirroring for high fault tolerance. There is some performance gain for Read accesses as two drives can be read in parallel. However, there is a performance loss for write access as the same data needs to be written on two drives.

Pros: Very high fault tolerance; slight gain in Read performance

Cons: 50 percent memory efficiency; loss in write performance due to processing overhead for writing two drives

RAID Level 2

This RAID level consists of bit-level striping with dedicated Hamming-code parity. All disk spindle rotation is synchronized and data is striped such that each sequential bit is on a different drive. This RAID level is not used by any of the commercially available systems.

RAID Level 3

This RAID level consists of byte-level striping with dedicated parity. All disk spindle rotation is synchronized and data is striped such that each sequential byte is on a different drive. Parity is calculated across corresponding bytes and stored on a dedicated parity drive. This RAID level is not commonly used in practice and is replaced by RAID Level 5.

RAID Level 4

This RAID level consists of block-level striping with dedicated parity. This RAID level is not commonly used in practice and is replaced by RAID Level 5.

RAID Level 5

This is one of the most popularly used RAID implementations in the industry. It uses striping to enhance the disk I/O performance. Moreover, it uses single parity (or data encoding) for fault tolerance. The write performance is lower than RAID 0 due to the overhead of parity calculation. This is extensively used in web applications that need fast read accesses with fault tolerance and rare write accesses.

RAID 5 distributes the parity blocks over multiple disks, thereby reducing the overhead caused by having a single parity disk. In a dedicated parity disk, a bottleneck is created as the parity disk can be written to only after the parity is calculated.

Pros: High Read/Write performance; Fault Tolerance – can survive up to 1 disk failure at a time.

Cons: Lower memory efficiency than RAID 0 due to the parity disk, but more than RAID 1.

RAID Level 6

This is a double parity RAID offering more fault tolerance than RAID 5 at the cost of an additional disk and processing power. With reducing costs of disk drives and increasing processing power of RAID controllers, RAID 6 is gaining popularity. This RAID implementation allows recovery from two simultaneous disk failures. The possibility of simultaneous failure of more than two disks is very low making this a RAID of choice for applications needing high fault tolerance.

The double parity may be implemented either by generating a diagonal polynomial for calculating second parity or by using an ECC algorithm like Reed-Solomon. If two disks fail at the same time, recovery is possible with the help of two parity blocks.

Pros: High Read/Write performance; Fault Tolerance – can survive two simultaneous disk failures.

Cons: Lower memory efficiency than RAID 5; significantly higher processing power required for double parity calculation.

RAID Controller Architecture

Figure 8 presents a generic RAID Storage system architecture. The data from the servers connected on the network is received by a switch which directs this data to different storage boxes. The processor in the storage box also interacts with multiple storage boxes through the switch to enable virtualization. The data received by the processor is sent to the RAID controller. The RAID controller may be implemented in software or as hardware RAID on Chip (RoC). A software implementation of RAID is typically done on the motherboard chipset. The RAID controller implements the RAID level intended in the application and interacts with the cache memory and disk drives directly for transfer of data.

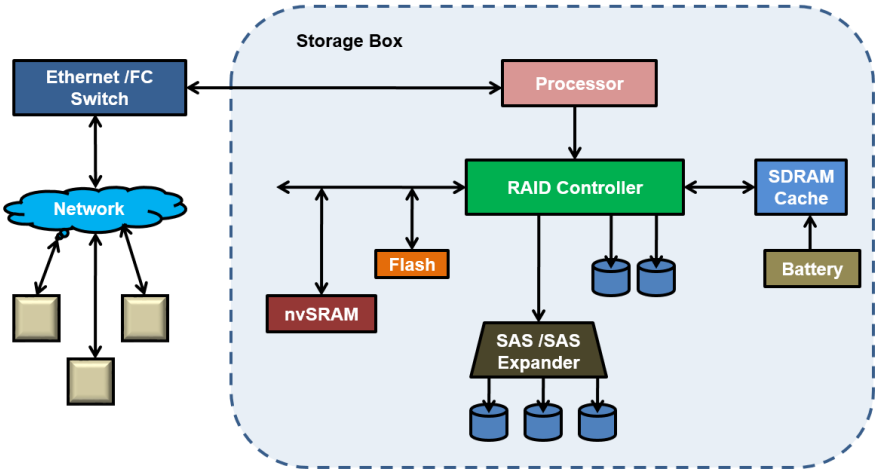

Figure 8. RAID Storage Architecture

Cache

The RAID controller has direct access to the cache memory that enables fast read/write access to the storage system. The cache is used to write the data in transition. The RAID system uses a cache to speed up the apparent I/O performance of the storage system so that the host processor is not kept busy. In modern RAID systems, the size of these cache memories is very large and high-speed DRAMs are typically used.

Read Cache

A read cache memory is used to speed up the read process by predictive caching of the expected reads. This cache memory helps in reducing the disk drive seek latency for read accesses by the processor.

Write Cache

Write cache is one of the most important components of a high performance RAID system. In a write cache based RAID system, the writes access from the host is acknowledged as soon as the data is received and stored in the cache. Since the disk drive access is not required, the throughput of a write operation significantly increases. The data is transferred from the cache to the disk drives by the RAID controller in background. This operation is transparent to the host which assumes that the write operation is complete as soon as it received the acknowledgement from the RAID controller. A RAID array may operate in two different modes:

Write Through Mode: In write through mode, the data sent by the host is directly written to the disks by the RAID system as the cache is bypassed. The Host waits for the RAID controller to signal completion of data write before sending the next data. Though the write throughput in this mode is significantly lower, it ensures integrity of data as the host is only acknowledged when the data is committed to the disk

Write Back Mode: In write back mode, the RAID controller writes data from the host to the cache memory and acknowledges write completion to the host. The host is free to perform other tasks while the RAID controller transfers the data from the write cache to the disk drives. Although this approach significantly increases write performance, the actual transfer of data to the disks is transparent to the host. This raises a potential risk to the data integrity in case a power failure occurs after host was acknowledged but before the data was committed to the disk. There are several ways to improve the data integrity while using the write back cache, most importantly to have a battery backed cache. A write journal based system is very effective in recovering from such power loss. This is discussed in detail in the next section.

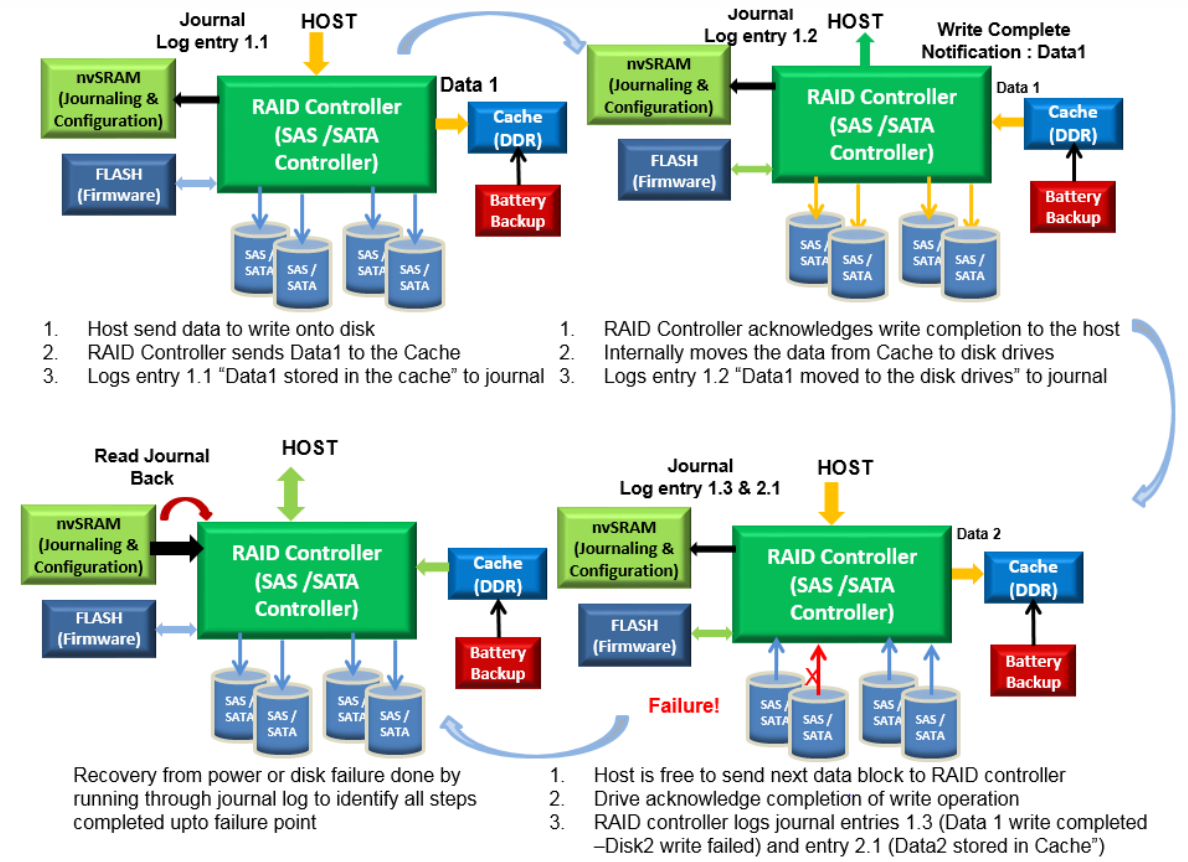

Figure 9. RAID Controller – Flow of Data

Write Journal

A write journal or write-in-progress record is used in 'Cache Write Back' Mode to keep track of the ongoing write transactions. A write-back cache improves the performance significantly, but it comes with an increased risk to the data integrity of the system. In write-back cache mode, the RAID controller does not wait for the data to be committed to the disk before acknowledging the host about completion of Write. The data is transferred from the cache to the disk drives internally and this movement of data is transparent to the HOST. After it receives acknowledgement from the RAID controller, the HOST assumes that the data sent has been securely stored to the physical disk drives. However, power failure, disk failure, or any other failure that inhibits the completion of data movement from the cache to disk drives may cause system failure.

Consider a situation where the storage system acknowledges the host after writing data into the cache, but the power goes off before the system transfers the data from cache to the physical drives (as shown in Figure 9). The data stored is usually secured using either a capacitor or battery backup that retains the contents of the cache even during power failure. On the next power up, the storage system has to track back each transaction to find which blocks of data were not written into the physical drives. Therefore, even with battery backed cache, the system recovery times may be huge as the system needs to be checked for the exact failing point before recovery can begin.

To avoid such recovery delays, many storage systems log each transaction of data from the host to the cache and from cache to the disk in a nonvolatile circular buffer. When power goes off, this log can be played back to precisely identify the state of the storage system before failure. This strategy is called write-in-progress record or 'Write Journaling'.

Journaling Concept

The concept of Journaling is not new to the world of computer science. Journaling File Systems (JFS) have been used with huge success in past. The following is the definition of JFS from Wikipedia:

"A journaling file system is a file system that logs changes to a journal (usually a circular log in a dedicated area) before committing them to the main file system. Such file systems are less likely to become corrupted in the event of power failure or system crash"

This definition is very close to how write-journaling is implemented in RAID systems. The keywords to be noted are:

Dedicated memory area

A dedicated memory area is recommended because it prevents corruption of the journaling memory at the same time as corruption of main file system due to power loss or system crash. This is the reason why RAID systems use a separate nonvolatile memory for storing write-journal information.

Less prone to power failure or system crash

It is not very difficult to imagine why journaling makes the system more robust against system crash or power failure. RAID systems use the same principle to reduce the possibility of loss of data integrity due to power, disk, or system failure.

Database ACID Properties - Importance of Journaling

ACID is an acronym used in database terminology to describe the four important properties that a robust database must ensure – Atomicity, Consistency, Isolation, and Durability. Since, in most of the applications, RAID system is used for storing databases, it is very important for the ACID properties to be followed even in the hardware. Journaling is a simple way to ensure that a RAID system is compatible with at least three of the ACID properties. The four ACID properties and importance of journaling in achieving these in a RAID system are enumerated here:

Atomicity: Atomicity refers to the capability of a system to ensure that either none of the parts of a transaction fail or all of them fail. In a RAID system, a write-through cache mode ensures this by not acknowledging the data write complete unless the data is actually committed to the disk drives. However, in a write-back cache mode, after the RAID controller acknowledges receipt of data to the host (which implies that one part of the transaction was successful), it must ensure that the data is in fact moved from the cache and written to the disk drives (all parts of the transaction are successful). A write journal helps the system achieve this by making it possible to identify exactly which parts of the transactions failed and need to be completed after the system recovers from a failure.

Therefore, the concept of write journaling is very closely coupled to the atomicity of write operations in the write-back cache mode of the RAID systems.

Consistency: Consistency refers to the property of a system to allow only valid data to be written to the database. This implies that if, at some point, the operation performed violates the consistency rules, the entire transaction must be rolled back to make the database consistent again. A write journal allows easy identification of the sequence in which each operation was performed and therefore makes it easy to achieve consistency. With the knowledge of the sequence of events, all the operations of an invalid transaction can be rolled back.

Isolation: It is the property of a database system to isolate any concurrent operations. Any data in transitional state is not visible to the other processes. Write journaling ensures that the system is always aware of the state of any operation in transition. This makes it easy for the system to ensure that any transitional data is not used (not visible to any other operation).

Durability: Durability is the property of a database system to ensure that once a transaction has been acknowledged (or notified to be complete) it must be completed. This is a key problem solved by using write journaling in a write-back cache mode. In this mode, the RAID controller

acknowledges the completion of write to the host as soon as the data is sent by the host. This data is temporarily stored in the cache. A write journal ensures that the operation is complete even if the power goes down before the data is moved from cache to the disk drives.

A write journal enables easy implementation of the ACID properties to be followed by a database system. The ACID properties are usually handled at the software level – but a write journal ensures that the software has sufficient assistance at hardware level too to meet these requirements.

Note Although, a nonvolatile write journal makes the implementation of ACID properties in a database simpler, it is not a sufficient condition for database meeting ACID requirements. Care must be taken in hardware and software design to ensure that the ACID properties are met by the system.

Nonvolatile Write Journal in Parity based RAIDs

As discussed earlier, parity (or encoding) is used to recover the system from a single or multiple drive failure. A single parity scheme can help recover from a single drive failure. However, in a parity based RAID system, the parity block needs to be generated each time a new data block is written. Calculation of parity can be a three step process: read, calculate parity, and write parity. Therefore, the parity block may not be written at the same time as the data blocks. In such a system, there is a constant risk of data loss due to a single disk failure when the data blocks were written but the parity block was not or vice-versa. Although nonvolatile write journal is important for any RAID system, it becomes even more critical for a parity based RAID system (for example, RAID levels 5 or 6).

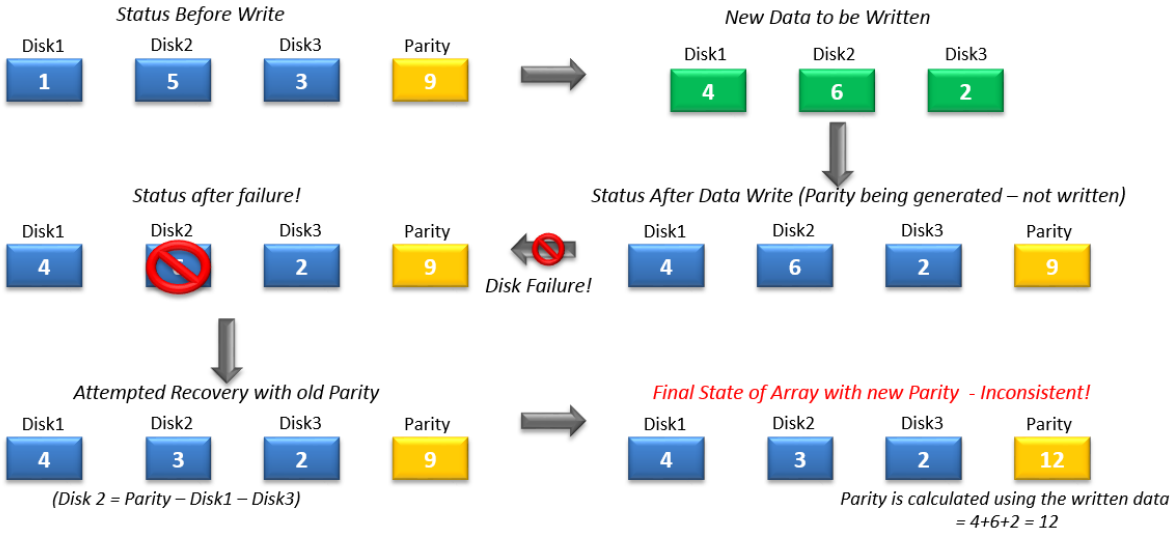

Figure 10. Possible Issues in Parity Based RAID Systems without Write Journaling

As demonstrated in Figure 10, the attempt to recover a failing disk with the help of parity leads to inconsistency of data if the parity block was not written to the disk when the failure occurred. The question that arises is the probability of occurrence of this condition. Each time a write is done to the disk, there is a parity to be generated, and if the disk fails, it may lead to inconsistency of data. Therefore, the probability of occurrence of this condition is equal to probability of disk failure when a write is going on. This is a huge number in a write-intensive application.

The nonvolatile RAM (NVRAM) used for write journaling records the completion of writes to each drive. Therefore, this record can be played to identify the operations that were not complete before the failure occurred. Once the RAID controller realizes that the parity block was not written to the disk drives, it can stall the recovery attempt till the writing of parity is completed. This is a simple solution to save the system from inconsistency and is widely used in the storage market today.

Choice of Memory Device for Write Journaling

Choosing the right nonvolatile memory type for write journaling is important for performance and reliability of RAID system. Though you are backing up the cache with a battery anyway, a part of the same cache memory cannot be used as the write journal because of the following reasons:

Efficiency: The speeds at which SDRAMs work today are enormous. In practice, too, large blocks of data are being written to the cache continuously at very high speed. However, only small blocks of data are being written to the journal memory and even these accesses need not be at speeds as high as DDRII and DDRIII. Using a part of cache as a write journal requires the system to stall the writing of Kbytes of data for the few bytes of journaling data to be written. This, in effect, leads to inefficient usage of cache bandwidth and impacts the speed of a RAID system.

Redundancy: Placing the journal memory in the write cache is like putting all your eggs in one basket. This creates the risk of losing both cache data and write journal at the same time.

Durability: Cache memories are typically battery backed SDRAMs. Usually, these have limited power down lifetime of 72 hours. A dedicated nonvolatile memory for write journaling ensures that even if the cache data is lost, the system can identify the status of operations before the power failure happened.

Availability: An independent access to the write journal allows the system to accept new data to be written from the host even when it is reading the write journal. This helps to identify the completed and pending writes to the disk and increases the system availability which is of prime importance to any storage system.

Choosing the Right NVRAM for Write Journaling

The perfect memory choice for write journaling is very critical for system performance and reliability. The ideal choice for a memory for this application is a nonvolatile memory which allows:

Fast Read/Write Access: For high performance of RAID systems, the NVRAM should be able to handle fast read and write accesses. A slow journal can potentially slow down the whole storage system and can become a burden or bottleneck of the system.

High Reliability: Data reliability of journal entries is important to attain the goals of atomicity, consistency, and more importantly, durability. This memory space is also used to store some configuration details and failure logs with time stamping. Therefore, highly reliable memory is a must for this application.

High Endurance: Data needs to be written and overwritten at a high pace in a journal. This makes it necessary for the journal memory to have very high endurance.

Cypress nvSRAM

nvSRAM is a high endurance, high speed reliable nonvolatile memory which has an SRAM interface. nvSRAM consists of SRAM cells with nonvolatile elements integrated into each cell. Data is read and written through the SRAM interface and when power goes down, the SRAM contents are transferred to the nonvolatile elements.

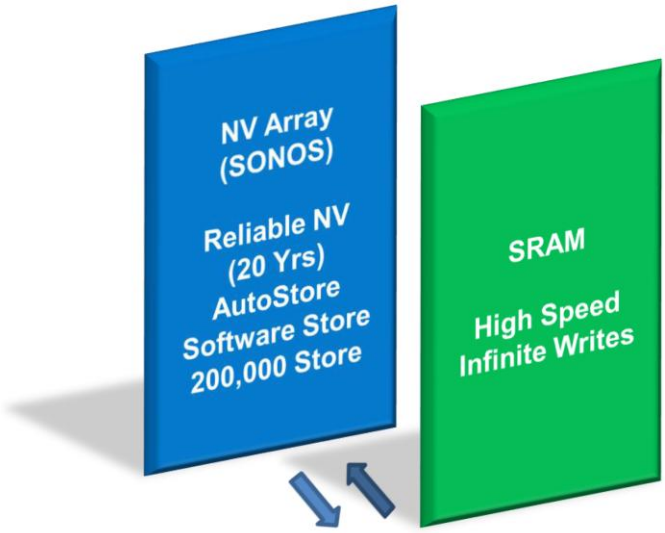

Figure 11. nvSRAM - Fast, Easy, Secure

The SRAM interface gives nvSRAM a very high speed read and write access (up to 20 ns asynchronous access and up to 108 MHz quad SPI access) and allows the nvSRAM to be infinitely written or read. The integrated SONOS based NV elements provide reliable data storage without the need of a battery to protect the SRAM contents. The endurance of nvSRAM is counted in terms of number of times power is lost. It is only when the power goes down that a store operation to NV elements takes place.

nvSRAM is the fastest nonvolatile Random Access Memory in the market today and allows infinite write operations. The data backup to nonvolatile memory is done automatically on power failure using energy from a small capacitor. The parallel transfer of data from SRAM to the integrated NV elements means that the STORE (transfer of SRAM data to nonvolatile elements) takes time equivalent to one write operation of EEPROM.

nvSRAM - The Ideal Choice for Write Journal

nvSRAM is the most commonly used nonvolatile technology for write journaling and with good reason. nvSRAM fulfils the requirements outlined for an ideal memory choice for the write journal:

Fast Access: nvSRAM is the fastest nonvolatile Random Access Memory in the industry with 20 ns read and writes accesses with asynchronous interface and 108 MHz with quad SPI. This allows RAID system designer to concentrate on more important aspects of the system design and not worry about managing a slow write journal.

Reliability: Silicon-Oxide-Nitride-Oxide-Silicon (SONOS) technology provides nvSRAM with unmatched reliability. Data retention of 20 years for the industrial temperature range is the highest among nonvolatile RAM technologies available today. This makes nvSRAM the most reliable nonvolatile technology and an ideal choice for write journaling application.

Unlimited Endurance: The SRAM interface of nvSRAM allows the write journal to be written infinitely without the worry of running out of endurance or wear-leveling. This takes away all the effort

required for implementing wear-leveling or counting for endurance cycles in software and helps the system designer focus on other issues.

Green (No Batteries): Unlike a battery backed SRAM, nvSRAM is a green technology with no battery involved. A small capacitor is used for transferring the contents from SRAM to the nonvolatile memory after which the nvSRAM can sustain itself without requiring any additional power.

Real Time Clock: Many RAID systems prefer to time stamp the write journal data. This typically requires an additional Real Time Clock (RTC) device increasing the total BOM cost and board area. While using nvSRAM, this is not a concern as nvSRAM comes with the option of integrated RTC. This not only reduces the BOM but also uses fewer pins for the RAID controller and frees up the resources required to interface with the external RTC device.

Summary

Most of the storage systems today use RAID for higher performance and data reliability. RAID systems use a fast SDRAM as a write-back cache for high performance. Although it increases performance, a write-back cache increases the risk on the system data reliability. A write journal is a nonvolatile random access memory used in a RAID system for keeping record of the ongoing transactions and helps increase the system reliability and while incorporating ACID properties expected from a database system. Cypress's nvSRAM, with its infinite read and write endurance, high reliability, and high performance write and read access is the memory of choice for the write journaling application.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com