White Paper

EMC VNXe3200 MCx™

Multicore Everything™

Abstract

This white paper explains the EMC VNXe3200 MCx technology and its main components: Multicore RAID, Multicore Cache, and Multicore FAST Cache. Together, they offer new VNXe features such as Multicore Scalability, Adaptive Cache, Permanent Spares, and many others.

January 2017

EMC²

Part Number H13725.1

Table of Contents

Introduction

A modern storage system must deliver higher IOPS than ever before, a need driven by modern multicore/multi-socket server architectures. Today's storage systems must be able to serve more host-based CPU cores in the storage environment. Moore's Law—higher granularity silicon geometry and 3D chip design for the future servers—promises to sustain that trend. A storage system deployed today and kept on the data center floor for five years must be ready to deliver 10X the transactions in its service life. To avoid storage bottlenecks, the core count on the storage side must keep pace with the continually climbing CPU core count on the host side.

Capacity is also a component that continues to climb. A few years ago, a typical midrange storage system only needed to store 10-20TB of data. Today, midrange systems' storage capacity has climbed to 100-200TB. Storing more data has been enabled by driving up the capacity per disk slot. Disk capacities grow in step with the increased area density of the magnetic platters used in mechanical disk drives. In fact, the disk drive area density continues to increase year over year. However, access speed has not been growing at the corresponding rate, leading to a loss in access density. As a result, disk rebuild times are now prohibitively high.

The increase in capacity places new demands on the storage system's cache. As system capacities grow, the system cache must follow suit. As a rule of thumb, a good cache target should be 0.1 percent of the total storage capacity. For example, at 20TBs of capacity, a 0.1 percent cache target is 20GB of DRAM. Such large DRAM sizes are both costly to acquire and to cool. Fortunately, Flash-based cache is both cost-effective and fast enough to provide an excellent DRAM alternative.

To take full advantage of modern CPU designs and developments in Flash storage while continuing to deliver increased value for EMC's midrange storage customers, EMC developed the MCx™ platform—the latest in modern multi-core/multi-socket storage technology. Several years in development, MCx leads in horizontal core scaling, which targets data management and storage applications. EMC VNXe3200 systems with MCx deliver unprecedented performance and platform efficiency. This white paper discusses some of the key innovations of MCx as it relates to EMC's VNXe3200 storage systems.

Audience

This white paper is intended for EMC customers, partners, and employees who are interested in learning about the MCx functionality in the EMC VNXe3200 storage system. It assumes that the reader has a basic understanding of storage and SAN principles.

Terminology

Multicore Cache – MCx software component that optimizes a Storage Processor's DRAM to increase host write and read performance. Designed to effectively scale across multiple CPU cores, Multicore Cache ties all other MCx components together and provides write caching for all VNXe software layers.

Multicore FAST Cache – MCx component that identifies frequently accessed blocks of data and copies them from a storage pool's spinning drives to Flash drives in Multicore FAST Cache, providing Flash drive performance to highly accessed data in your storage pool.

Multicore RAID – MCx component that defines, manages, creates, and maintains RAID protection for any created storage pools. Designed to effectively scale across multiple CPU cores.

Storage Pool – Collection of disk drives configured with a particular storage profile. The storage profile defines the type of disks used to provide storage and the type of RAID configured on the disks. The storage pool's configuration defines the number of disks and quantity of storage associated with the pool.

Storage Processor – Hardware component that performs VNXe storage operations such as creating, managing, and monitoring storage resources as well as handling I/O from hosts.

Unisphere CLI (UEMCLI) – Command-line interface for managing VNXe storage systems.

Unisphere for VNXe – Web-based user interface for managing VNXe storage systems.

MCx Overview

Developed with many goals in mind, MCx's most significant objective is CPU core scaling. MCx takes full advantage of the multiple CPU cores that Intel's microarchitecture offers. CPU scaling has become a fundamental requirement for further technological growth. As the number of CPU cores and processor speeds grow in the future, the storage system's software stack must be able to scale with additional processing power.

MCx changes the way the VNXe3200 handles threads and thread distribution. Scheduling CPU-cores in parallel does not come free. Consider Amdahl's law[1], "*the speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program.*"

No system can use all of its CPU-cores linearly. Four CPU-cores do not complete a program four times faster than a single core. Consequently, spreading threads across other multiple CPU-cores causes the scheduling CPU-core to spend time managing the distribution. But Amdahl's law does not prohibit loading the cores on some other basis. For example, front-end Fibre Channel (FC) ports can be aligned with different cores to distribute work more effectively.

Goals

MCx is designed to achieve a number of goals, each of which this paper addresses:

- Scale on multi-processor architectures
- Improve performance and scalability
- Optimize memory efficiency

Stack Components

As a design rule for MCx, EMC developed a Multicore Everything mindset. The new way of thinking ensures that a technological breakthrough requires a new software foundation. EMC started by reinventing its core components:

- Multicore Cache
- Multicore FAST Cache
- Multicore RAID
- Multi-Path Communication Manager Interface

Core Affinity

MCx has a concept of a preferred core. Every front-end and backend port has a preferred and alternating core assignment. The system keeps front-end host requests serviced on the same core they originated, to avoid the adverse performance impact of swapping cache and context between CPU-cores. On the backend, every core has access to every drive, and servicing requests to the drives do not require switching to another CPU. When a preferred core is busy, an alternate core is used.

[1] Amdahl's law, "Validity of the single processor approach to achieving large scale computing capabilities", 1967, http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf

Less-busy cores process non-host I/O, or the I/O generated by the storage system itself. Backend I/O operations such as disk rebuilds, proactive copies, Sniff Verify, and disk zeroing, do not impede Host I/O.

MCx leverages the CPU processing ability more effectively, directly increasing performance. MCx also enables scaling performance to much higher levels. As port usage grows and I/O load increases, MCx automatically distributes processing to all available cores. Introducing additional workload spread across all front-end ports on the same system shows the scaling power of MCx.

Multicore Cache

The VNXe3200 system's Multicore Cache, also known as SP Cache, is an MCx software component that optimizes a storage processor's DRAM to increase host write and read performance. Multicore Cache was designed to effectively scale across multiple CPU cores. The cache ties all other components together. Multicore Cache provides write caching for all VNXe software layers.

Multicore Cache features:

- Multicore CPU scaling
- Adaptive and automatic performance self-tuning

Multicore Cache operates in one of three states, shown in **Table 1**. In the Enabled state, Multicore Cache is fully functional and operates normally. In the Disabled state, Read Cache is still enabled but Write Cache is disabled and operates in a Write-Through mode. In the Dumping state, the VNXe3200 system is placed in a low-power state and all dirty pages in cache are de-staged to the internal mSATA device on each Storage Processor.

Table 1 – Multicore Cache States

Multicore Cache State	Read Cache State	Write Cache State
Enabled	Enabled	Enabled
Disabled	Enabled	Disabled (Write-Through Mode)
Dumping	Enabled	Disabled

More information can be found in the **Disabled State**, **Dumping State**, and **Write-Through** sections. The **Failure Handling** section provides a full list of hardware faults and their impact on the state of Multicore Cache.

Adaptive Cache

The Multicore Cache space is shared for writes (1) and reads (2), as shown in **Figure 1**. Instead of flushing a dirty cache page to disk, the page is copied to disk (3), and therefore it remains in memory and is not discarded. This way, the page can be reused again by the host (4) and (5), which improves performance. Data is ultimately expelled from cache (7), which frees the page for other workloads.

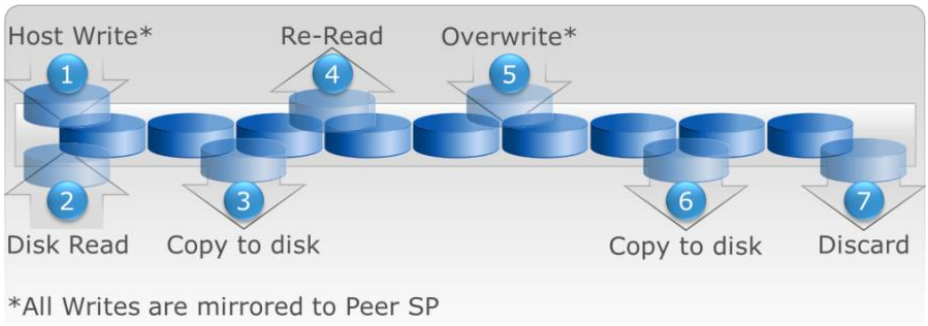


Figure 1 – Adaptive Cache

Mirroring is important—VNXe guarantees data durability the moment the write I/O hits the cache. A dirty page is always copied to the peer SP cache before the VNXe sends the acknowledgement back to the host/client. If one SP fails, the remaining SP has a copy of the data. For the peer SP, the mirrored page is the same as any other dirty page in cache, except for its ownership. Both SPs keep track of the pages they own and the pages owned by the peer. Page ownership changes if one of the SPs becomes unavailable (for example, if it is placed into Service Mode, reboots, and so on.). Each SP is responsible for cleaning its own dirty pages to the backend, clearing the dirty flag in both caches.

NOTE: Multicore Cache always mirrors every committed write I/O to the peer SP when Write Cache is enabled, as long as the peer SP is operational.

Multicore Cache keeps very detailed statistics on every cache page. It uses the concept of a page aging, or continuously growing. Every successful hit decreases the page's age and thus prolongs its life in cache. As a result, recent events are considered more significant. For example, what has happened in the last hour is much more important than what happened last month.

Storage pools also fully benefit from all Multicore Cache innovations. Storage pools are built from a set of underlying RAID groups, known as private RAID groups. The system, rather than the user, manages every private RAID group. This is fundamental to the design of storage pools. Multicore Cache monitors and manages traffic to the underlying RAID groups in storage pools.

Dynamic Watermarks

Instead of keeping track of just the dirty cache pages, Multicore Cache also monitors the Logical Block Address (LBA) corresponding to these pages' reference, as well as the demand that hosts put on Multicore Cache and the underlying RAID groups. The system constantly evaluates the effectiveness of cache for specific workloads. Buffering write I/Os in cache is effective for short activity bursts, but is counterproductive for workloads that tend to occupy the entire cache space.

Multicore Cache measures the rate of incoming I/Os and monitors underlying RAID groups to predict its effectiveness for workloads. A workload that sends write I/Os to a RAID group with many Dirty Pages waiting to be written to disk is likely to use more cache resources. The system evaluates the difference between the rate of incoming writes and the rate of successful RAID group page copies.

This evaluation allows Multicore Cache to react dynamically to workload changes:

- Workloads exhibiting signs of possibly overwhelming the cache can be subjected to write throttling
- Temporary short bursts can be fully buffered by cache
- Small block random writes can be coalesced for improved disk de-staging

Pre-Cleaning Age

The Pre-Cleaning Age is the maximum time allowed for a cache page to remain dirty. The Cache Page Age is the length of time that the page has been dirty. The Pre-Cleaning Age value is one of the most important Multicore Cache statistics.

The Pre-Cleaning Age is a dynamic value generated by Multicore Cache on per RAID group basis. It directly depends on the:

- Rate of incoming I/O that addresses the RAID group
- Rate of successful page flushes to the RAID group
- Number of available/free cache pages
- Effectiveness of the workload coalescing

To determine the correct time for a cache dirty page to be flushed to disk, Multicore Cache compares its age with the dynamic value of the Pre-Cleaning Age. If the cache page is younger than the Pre-Cleaning Age, everything is normal. If the cache page is the same age as the Pre-Cleaning Age, it is time to be flushed. If the cache page age is older than its Pre-Cleaning Age, the Multicore Cache can increase outstanding I/Os to the disks or enable write throttling.

A well-tuned Pre-Cleaning Age value is a key to effective cache management and to ultimate system performance. Multicore Cache is continuously adjusting the Pre-Cleaning Age based on the current workload being handled by the cache. Multicore

Cache protects its resources from being over consumed by a single workload addressing an overwhelmed RAID group.

Write Throttling

The Write Throttling feature of Multicore Cache provides the ability to auto-adjust incoming and outgoing I/Os per RAID group. This allows the cache the time needed to adjust the Pre-Cleaning Age for a given RAID group. When the age of the oldest dirty cache page is older than a currently set Pre-Cleaning Age value, the probability of cache being overrun by a write-intensive workload increases.

Multicore Cache provides the destination RAID group with the time needed to process an increased flushing load by delaying host I/O acknowledgements, thus slowing incoming write I/Os (Figure 2).

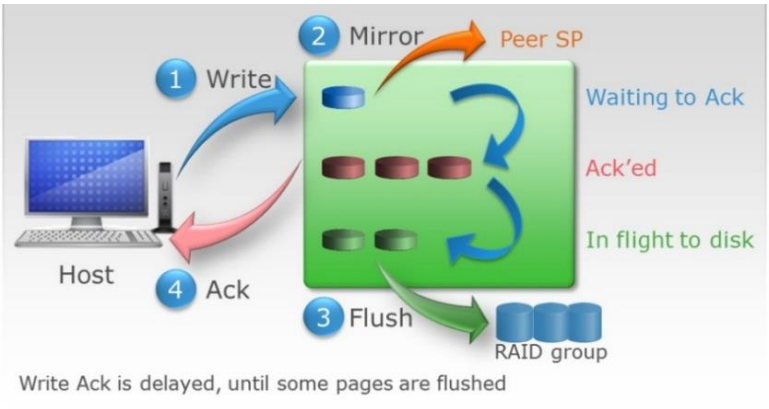


Figure 2 – Multicore Cache Write Throttling

Write Throttling continues until the rate of incoming data is equated with the abilities of the underlying RAID group and the Pre-Cleaning Age value changes to match the workload.

Write Throttling has the following goals:

- Protect the system resources from being monopolized by a single workload
- Equate write-intensive workloads with the disk speed
- Prevent write-intensive workloads from consuming cache pages needed for other workloads by learning the correct Pre-Cleaning Age value

Write Throttling does not continue indefinitely. It stops when the RAID group's rate of page flushing levels with the rate of incoming I/O, and the associated dirty pages stay in cache for equal or less time than the Pre-Cleaning Age value regulates.

As stated in the previous section, the Pre-Cleaning Age value is continuously adjusted. Using small corrections, Multicore Cache dynamically reacts to the workload changes. If the workload write activity subsides, the Pre-Cleaning Age value may increase. If the activity grows, Write Throttling can decrease the Pre-Cleaning Age value.

Write-Through

Multicore Cache read cache is always enabled, whereas write cache has two modes of operation:

- Write Cache enabled
- Write Cache disabled

When write cache is disabled, it operates in a Write-Through mode, which is a protective measure that occurs when there is a system failure that prevents write cache from being enabled. Write I/Os are not acknowledged by the VNXe system until they are successfully written to disk. There is also no mirroring with the peer SP.

When the VNXe system is operating in Write-Through mode, Multicore Cache is still used to receive the host I/O, however, the I/O is immediately written to disk. Once the I/O has been written to the disk, the VNXe system will send the acknowledgement back to the host. The I/O acknowledgement is sent to the host when the flush completes successfully.

Write Coalescing

For spinning drives, Multicore Cache offers substantial performance gains when writing I/Os to adjacent LBAs. For that purpose, Multicore Cache uses Write Coalescing, which groups LBA contiguous or neighbor pages together. The pages are organized in the sequential order, so it is easier for the drive to write quickly.

Page Size

Multicore Cache has two separate page sizes. Physical Page, or Cache Page, has a size of 8KB. The second page type is known as Logical Page, and its size is 64KB. One logical page contains one to eight physical pages. Multicore Cache uses logical pages within its algorithms (Figure 3).

Figure 3 – Multicore Cache Logical vs. Physical Page Size

Multicore Cache keeps the statistics of the number of allocated logical pages and dirty logical pages. Typically, only some physical pages are dirty in a set of logical pages. To get the Total Cache Dirty Size metric, Multicore Cache makes an approximation of that ratio based on averaging the write I/O load. Each Storage Processor makes its own calculation, and as a result, the Total Cache Dirty Size graph on the System Performance page might show different numbers for each SP.

Prefetching

Prefetching is a mechanism that proactively loads adjacent data into the cache based on data the host requests. It occurs in anticipation of the future read requests to reduce latency.

Spinning drives employ the concept of Seek Distance, the distance the magnetic head must travel to the needed area of the disk. Reading extra contiguous data is faster than repositioning the head to another part of the disk. If the prefetched data is used, then transferring additional data on the original read is cost-effective until the transfer time dominates the total I/O time.

To keep prefetching cost-effective, Multicore Cache pays attention to the following read-related criteria:

- Whether a prefetch has been valuable
- Whether the page was loaded quickly enough
- Showing unaccounted disk I/O wait
- Current decaying average read hits and read misses

Multicore Cache looks at the historical cache misses with the prior page in cache. This analysis helps Multicore Cache determine whether prefetching would have been beneficial had it been executed at the time of the prior page (Figure 4).

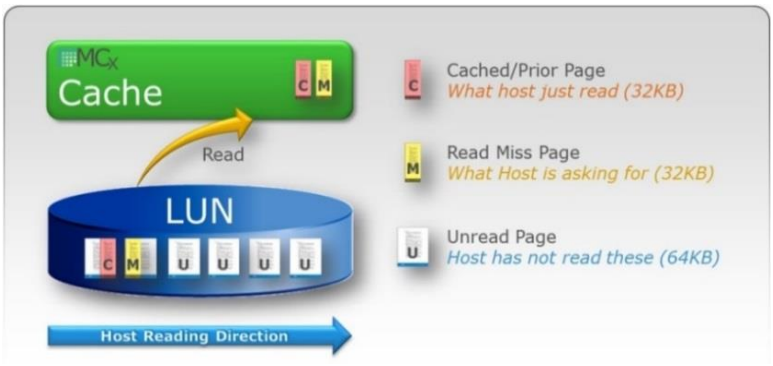


Figure 4 – Read Miss

The graphic shows a classic case when using a prefetch would have been beneficial. Even though the host is asking for less than a full logical page of data, the sequential aspect of the host read request suggests that it will ask for the rest of the page later. The data that the host read earlier (not shown in the graphic) might suggest whether the prefetch should have been more aggressive. Multicore Cache is looking for sequential workload and trying to identify it faster.

After identifying a sequential read workload and using the regular set of statistics (I/O size, queue depths, etc.), Multicore Cache triggers different prefetch speeds:

- Read Full page
- Read More (larger I/Os)
- Read Ahead (load before requested)

Read Full Page

With Read Full Page prefetching mode, an entire 64KB logical page is read into cache even if only a sub-portion has been requested by the host I/O (**Figure 5**). If a host requests regions that cross logical page boundaries, both pages are loaded (not depicted).

Figure 5 – Full Page Prefetch

Multicore Cache uses the decaying average read hit rate to calculate whether the read response time would be improved if the full page were read. The goal is to avoid full-page reads on small[2] sized random read workloads.

Read More

Depending on the workload, Multicore Cache may decide to mark a logical page with a Read More prefetch trigger. When a page with a Read More trigger is subsequently read by the host, Multicore Cache fetches another one.

Figure 6 – Read More Prefetch

Multicore Cache closely tracks the data prefetched with the read more trigger. These statistics ultimately answer the following questions:

- Was the prefetched page used by the host?
- What was the size of the host request that triggered a Read Miss?

Historical statistics of the LUN determine how aggressive Read More will be during the next Read Miss.

[2] In this case, small means smaller than a full page size, or 64KB.

Read Ahead

Read Ahead is an extension of the read more for more aggressive prefetching. Read Ahead is triggered when Multicore Cache has a reasonable expectation that the host is likely to address more data from the storage resource.

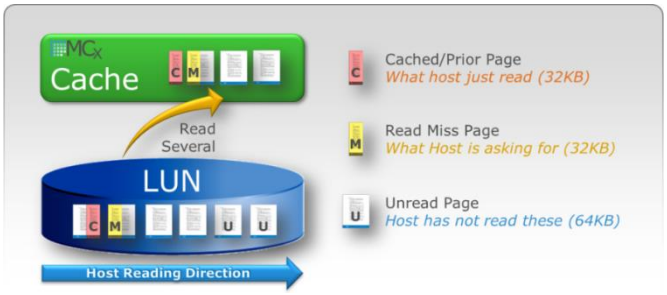


Figure 7 – Read Ahead Prefetch

Typically, read ahead prefetching occurs on larger sequential I/Os—for example, host reads with a 256KB I/O size. Please note that the host I/O size does not have to correlate with the disk I/O. Multicore Cache is loading 64KB of data, but it may choose to use a different I/O size on disk.

Multicore Cache Vaulting

The VNXe3200 protects Multicore Cache dirty pages with Backup Battery Units (BBUs) in the Storage Processors. In the event of a power fault, the batteries will keep the SPs powered long enough to be able to de-stage dirty cache.

When a hardware fault occurs, Multicore Cache has one of three states:

- Remains Enabled: fully functional
- Changes state to Disabled: all data writes go directly to disk
- Changes state to Dumping: the system is going down, all dirty pages must be de-staged to the vault

Disabled State

If Multicore Cache determines that it needs to enter the Disabled state, it cleans all Dirty Pages to disk. The cleaning is aggressive, but not at the expense of host I/O. After the cleaning is complete, Multicore Cache remains in a disabled state until the fault condition is corrected. In a disabled state, Multicore Cache performs as read cache, while write requests are handled in **Write-Through** mode.

Dumping State

If Multicore Cache determines that a dump is necessary, it flushes all dirty pages to the internal mSATA device on each Storage Processor. After the dump is complete, the SP continues the orderly shutdown. When Multicore Cache starts a dump, the Do Not Remove LEDs on both SPs are lit, providing a visual indication that the SPs should not be removed. The Do Not Remove LEDs are turned off when the dump completes. A full system reboot is required to recover the storage system after a dump.

Power Faults Table

The state of Multicore Cache based on different power faults are shown in Table 2.

Table 2 – Multicore Cache Fault Handling

Use Case	Description	Multicore Cache State
Single power failure, peer SP still running	Power is removed from one SP in a system	Enabled
Single SP over temperature	This SP or the power supply for this SP is reporting an over temperature condition	Enabled
Dual SP over temperature	An over temperature condition is reported by all SPs and power supplies	Enabled
Single DPE power supply fault	A single power supply is reported as faulted in the DPE	Enabled
Multiple power supply faults	Multiple power supplies are reported as faulted	Enabled
Single fan fault	A single fan has faulted	Enabled
Multiple fan faults	Two of the three fan modules are faulted on either side	Enabled
NDU without power supply firmware upgrade	Storage system software is being changed	Enabled
NDU with power supply firmware upgrade	Storage system software and the power supply firmware is being changed	Disabled
Single SP fault	A single SP is removed, rebooted, panics, or has a hardware fault	Enabled

Multicore FAST Cache

Multicore FAST Cache identifies frequently accessed blocks and copies them from spinning drives to Flash drives. Data is copied into Multicore FAST Cache when it has been repeatedly accessed. As with Multicore Cache, the data is flushed out of cache when it is no longer accessed as frequently as other data, per the Least Recently Used (LRU) Algorithm.

Multicore FAST Cache is a member of the MCx family, and is designed to:

- Scale to the number of available CPU cores
- Avoid inter-core traffic

Functional Design

Multicore FAST Cache works with file-level and block-level storage resources. For more information about the VNXe3200 Pool structure, refer to the **Introduction to the EMC VNXe3200 FAST Suite** white paper.

Memory Map (a component of Multicore FAST Cache) maintains the state of Multicore FAST Cache pages and their contents. It is a bitmap that coordinates the physical addresses for data drives with the corresponding Multicore FAST Cache chunk. The Memory Map is kept in two places: In Multicore Cache, and on the Flash drives used with Multicore FAST Cache.

Page Management

Multicore FAST Cache operates with a page size of 64KB. When a page is first copied to FAST Cache, it is marked as clean, meaning it is an exact copy of the data on the backend drives (in the storage pool). If it is modified by an incoming write I/O from Multicore Cache, the page becomes dirty. The dirty pages are copied back to the data disks in the storage pool, and become clean again. The copy to the disk is an asynchronous process, and happens sometime after the host acknowledgement. When necessary, the least recently used clean pages are overwritten by a new FAST Cache promotion from any FAST Cache-enabled storage pool resource.

Proactive Cleaner is a FAST Cache enhancement that copies dirty pages to their respective data drives and leaves them clean in FAST Cache. The process occurs when the FAST Cache has relatively low activity (**Figure 8**). If the rate of new page promotion and FAST Cache I/O activity is high, the cleaner has less opportunity to run. Consequently, the number of dirty pages grows or stays the same. If the activity is low, the cleaner runs more often, progressively reducing the amount of dirty data pages in FAST Cache.

Proactive cleaner uses the LRU algorithm to select which dirty pages to clean. Just like Multicore Cache, Multicore FAST Cache does not discard the page contents after a flush: a clean page is eligible for reuse by another copy.

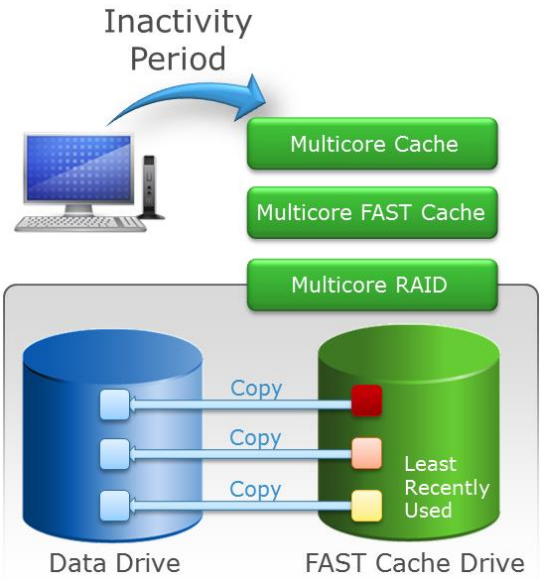


Figure 8 – FAST Cache Proactive Cleaner

Keeping the amount of dirty pages low benefits FAST Cache. If a new workload is presented to the storage system, FAST Cache can promote data at a faster rate, because it does not have to flush dirty pages to data drives prior to freeing older pages for reuse.

When FAST Cache identifies new promotions that require space, it first checks for available free or clean pages. If none are found, FAST Cache selects an appropriate amount of dirty pages using the LRU algorithm and copies them to data disks (flush operation) before reusing them for new promotions (**Figure 9**).

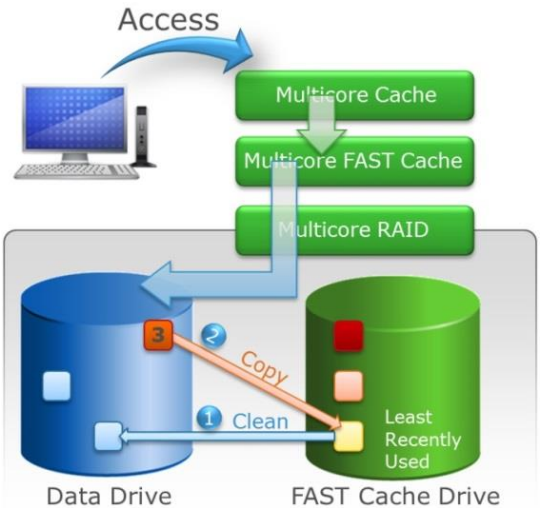


Figure 9 – Clean Before Copy

Minimal Page Count

Multicore FAST Cache prevents an individual storage resource from using all available pages at the expense of the other storage resources. A Minimal Page Count metric is used, which guarantees every storage resource a fair share of FAST Cache space.

The minimum page count is calculated by equally sharing 50 percent of all Multicore FAST Cache pages used by all storage resources in the system. The other 50 percent is freely available for any storage resource. If a storage resource does not use all of its minimal count pages, they remain freely available to other storage resources. The minimal count is further adjusted to be no more than half the size of the storage resource. The count is a dynamic number and is adjusted every time a storage resource is created or deleted.

Any new promotion into FAST Cache requires a clean or free page. FAST Cache chooses the pages so that every storage resource keeps its minimal page count for its most-used data. Consider an example with two LUNs (**Figure 10**). FAST Cache has eight pages total, and there are two LUNs present on the system. Each LUN minimal page count is set at two. (One-half of all eight pages is four, divided by two LUNs makes two.) LUN A promotions occupy six FAST Cache pages, leaving two pages free. Four out of six LUN A pages are hot; two other pages begin to cool down.

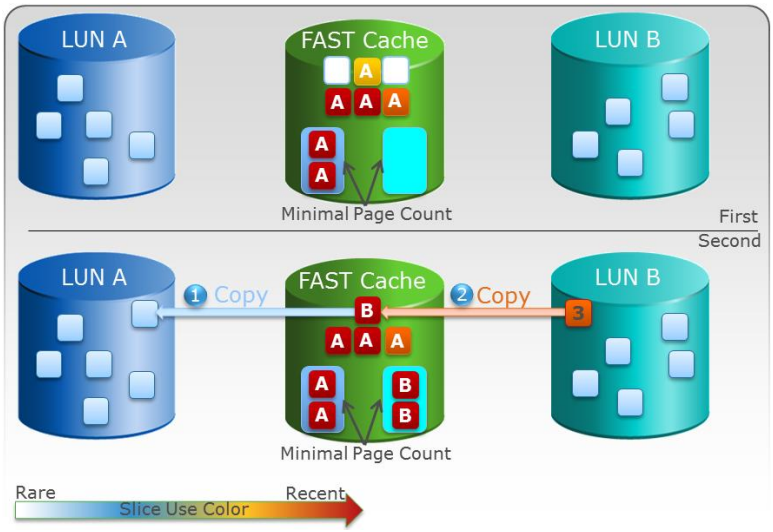


Figure 10 – FAST Cache Minimal Page Count

Then FAST Cache starts to promote LUN B's data. The first two data blocks from LUN B obtain the two free pages. A third promotion from LUN B causes FAST Cache to flush the least recently used page from the free-floating pool to LUN A and reuse the page with data from LUN B.

If additional data is promoted from LUN B, it will take up to three more pages from the free-floating pool, thus guaranteeing LUN A its fair share of minimal pages. If two new LUNs are added to this system, the minimal page count would become one page per LUN.

FAST Cache Writes

The FAST Cache memory map is located below Multicore Cache (Figure 11). The benefit is that MCx is quicker to acknowledge a host write.

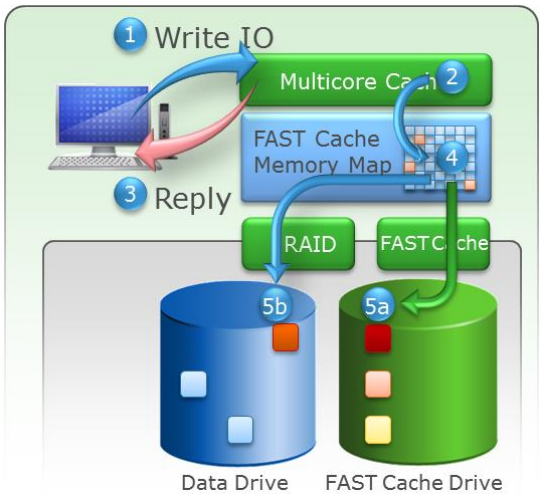


Figure 11 – FAST Cache Write

In the VNXe3200, the Write I/O (1) enters Multicore Cache first (2). An acknowledgement is immediately sent to the host (3). When Multicore Cache is ready to copy the data to the drives, it checks the FAST Cache memory map (4), and depending whether the data belongs to FAST Cache or not, it sends the data to the appropriate location: FAST Cache (5a) or data drive (5b).

MCx acknowledges the write I/O faster and the data spends more time in RAM. This combination significantly improves overall system performance. If Multicore Cache's write function is disabled, the incoming write acknowledgement is delayed until the I/O is saved either to the Multicore FAST Cache Flash drive, or to the data drive.

FAST Cache Reads

Read operations have also changed with MCx, but the overall effect is less noticeable by the hosts (Figure 12).

In the VNXe3200, the Read I/O (1) first enters Multicore Cache (2). A read hit condition causes an immediate reply to the host. A read miss condition results in the FAST Cache memory map being checked (3). If a FAST Cache record is found, the data is read from the FAST Cache drive (4a), or from the data drive (4b) if the record does not exist. The data is loaded to Multicore Cache, and a reply is sent to the host (5).

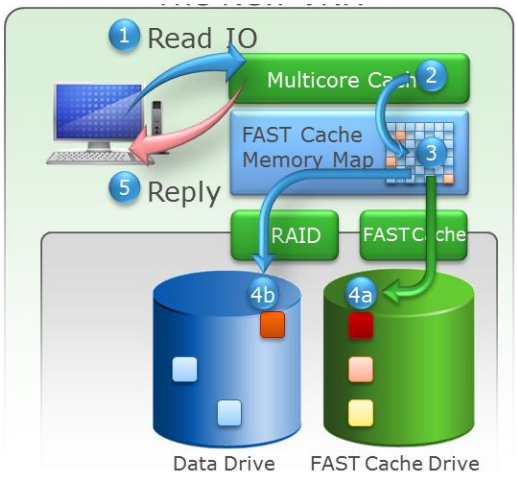


Figure 12 – FAST Cache Read

Interactions with Multicore Cache

Multicore FAST Cache and Multicore Cache are tightly integrated with one another. Both drivers perform specific functions and supplement each other. Multicore Cache shields Multicore FAST Cache from high-frequency access patterns: DRAM is quicker than Flash storage, and therefore better suits high-frequency access patterns.

Multicore Cache has native optimizations for sequential I/O: prefetching, coalescing, write throttling, and so on. Multicore FAST Cache directly benefits from this intelligence to concentrate on its own ideal workload — small random I/O.

Multicore Cache and Multicore FAST Cache each have their own strengths, and they allow each other to service I/O in the most efficient way. This improves the overall system performance and simplifies management.

Destruction of FAST Cache

MCx FAST Cache can be destroyed online. When ordered to be destroyed, FAST Cache immediately changes its state to Disabling. It stops all new data copies, and starts the flushing process, which expunges all pages from FAST Cache Flash storage. While the flushing process runs, Multicore FAST Cache still services hosts I/O for the pages that remain in cache.

The flushing process is not immediate, and it may take several minutes. The larger the FAST Cache, the more time it takes to flush all the data from it. After all pages are expunged from Flash storage, the mirrored pairs are destroyed, and the Flash drives change their state to unused.

Failure Handling

Multicore FAST Cache Flash storage is based on a common RAID group, and is subject to all Multicore RAID recovery processes, such as rebuild logging, write journaling, and so on.

Single Disk Failure

If one of the Multicore FAST Cache Flash drives fails, the corresponding RAID1 group becomes degraded. The pages that belong to this RAID1 group are flushed—clean pages are discarded and dirty pages are written back to the user LUNs. New hot pages from any user LUNs are promoted to the other remaining[3] Multicore FAST Cache Flash RAID devices.

Multicore RAID attempts to spare the failed drive. Once sparing completes, the RAID pair is put back into service. For example, new promotions into Multicore FAST Cache are allowed on that device again.

If all FAST Cache RAID pairs become degraded, the system disables FAST Cache and starts a cleaning process.

Multicore FAST Cache Configuration Options

Multicore FAST Cache forms RAID 1 (1+1) pairs based on the number of drives provided[4]. The maximum capacity that can be configured in Multicore FAST Cache for the VNXe3200 is 400GB[5]. The table below lists the different capacities and available configurations on the VNXe3200.

Table 3 – VNXe3200 FAST Cache Configurations

Multicore FAST Cache capacity	Available drive configurations[6]
100GB	2x 100GB Flash drives
200GB	4x 100GB Flash drives
	2x 200GB Flash drives
300GB	6x 100GB Flash drives
400GB	8x 100GB Flash drives
	4x 200GB Flash drives

NOTE: FAST Cache cannot be dynamically adjusted in the VNXe3200 system. In order to adjust the size of FAST Cache, you need to first destroy the current configuration and then recreate the FAST Cache configuration as needed.

[3] This will not be true if FAST Cache only has a single pair of assigned Flash drives, or if all pairs are degraded.
[4] Because Multicore FAST Cache creates RAID 1 mirrored pairs, an even number of drives must be available for use. Multicore FAST Cache configurations can only contain FAST Cache Optimized Flash drives.
[5] 400GB Multicore FAST Cache configurations are available in VNXe Operating Environment 3.1.0.x and later.
[6] You cannot mix drive sizes in a Multicore FAST Cache configuration.

Multicore RAID

Multicore RAID is the MCx component that defines, manages, creates, and maintains VNXe RAID protection for any created storage pools. The more cores the physical Storage Processor has, the more processing power is available for the RAID engine, as Multicore RAID operations are not isolated to a single core. Multicore RAID includes significant improvements—such as Permanent Sparing, Drive Mobility, Hot Spare Policy, and Improved RAID durability—which are discussed below.

Permanent Sparing

Hot Sparing, or Sparing, is the process of rebuilding a failed drive's data onto a system-selected compatible drive. Multicore RAID allows the permanent replacement of failed drives with spare (or unused) drives. When a drive fails, Multicore RAID spares it to a suitable unused drive, and that new drive becomes part of the storage pool's RAID configuration. When a failed drive is replaced, the new drive becomes unused and available to be a future spare.

Multicore RAID revolutionizes the concept of how drives are identified. Rather than using physical location as part of the drive ID, every drive is defined as a Virtual Drive and has its own serial number.

As shown in **Figure 13**, a drive from a given storage pool (red, for instance) has failed (or is replaced by a Proactive Copy mechanism) and is being spared. The replacement drive becomes a member of the storage pool, and the Multicore RAID engine rebuilds the drive. After the rebuild, the storage pool will operate normally.

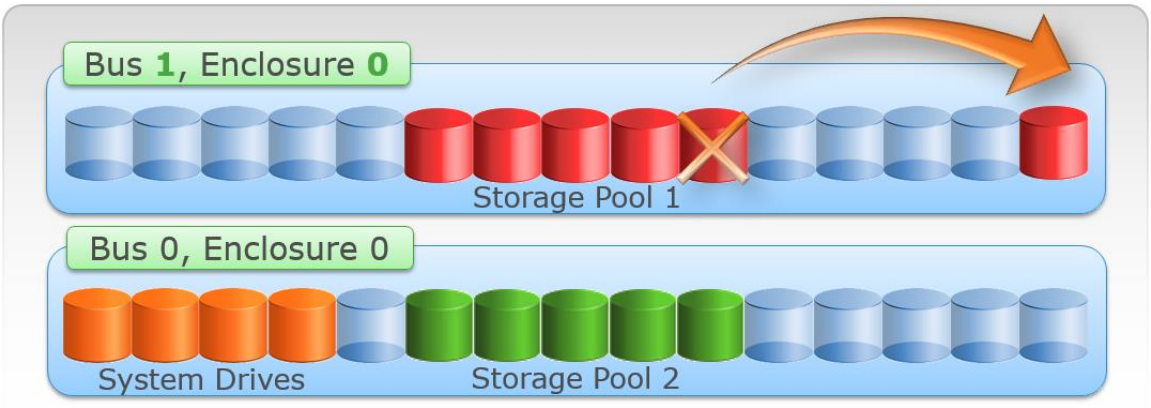


Figure 13 – Permanent Sparing

Drive Mobility

Drive Mobility, also known as Portable Drives, allows users to move VNXe drives within the storage system. Drive Mobility is very closely related to the Permanent Sparing feature. Multicore RAID allows physically moving drives and entire DAEs inside the same frame from their existing locations to other slots or buses.

Note: Drives do not retain data when moved between VNXe3200 systems. Every system keeps track of its own drives. All new and unknown drives are automatically formatted and zeroed.

Drives can be moved to any slot within the storage system, as long as they fit within the available space. **Figure 14** shows a red storage pool that has two drives moved to new locations, including a different enclosure.

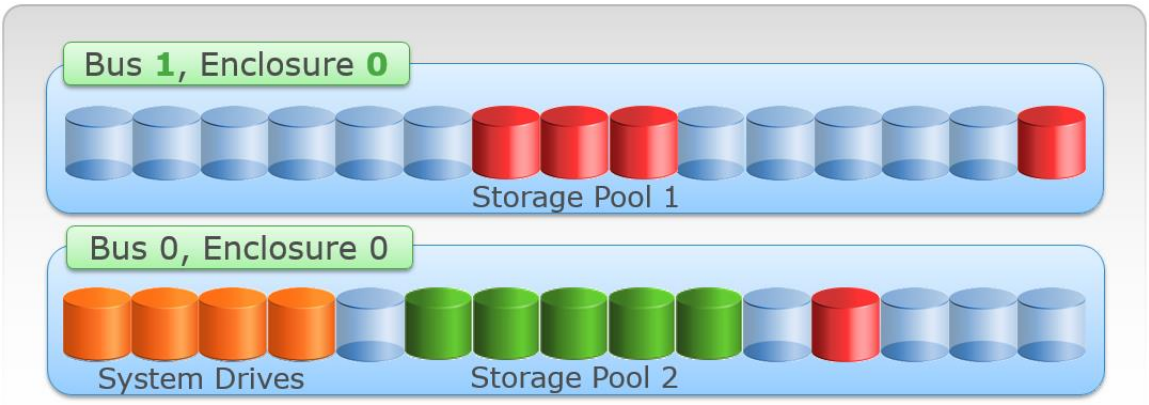


Figure 14 – Drive Mobility

Multicore RAID allows online drive movement. Storage pool data continues to be available after the drive is pulled out. When a drive becomes unavailable, such as being pulled out of a slot, MCx starts a 5-minute counter. In this instance, drive sparing is invoked only after the 5-minute interval passes. See the **Spare Drives** section for more details.

Pulling any active drive out of the storage system degrades its storage pool. If a storage pool is built with N+1 protection and one drive is removed, the +1 protection is no longer available when the drive is removed. If another drive fails while the drive is out of the system, the storage pool experiences a Data Unavailable event and becomes faulted.

Note: Move drives within the VNXe3200 system with caution.

After the drive is reinserted into the system, the storage pool returns to a healthy, operational state and no data is lost.

DAE Re-Cabling

Multicore RAID drive mobility also optimizes the availability of backend buses. A benefit of Drive Mobility is that re-cabling backend buses on the DAEs can now occur with all data preserved. Users can simply power the storage system down, cable DAEs as needed, and power the system back up. Upon power-up, the system can easily locate the drives by surveying their serial numbers, validate all storage pool members, and continue operations.

Some customers like to rearrange their storage pools for various reasons, such as adding more DAEs. Although it is possible to rearrange all drives while the storage system is up and running, it may be safer and faster to do mass-rearrangements when the system is shut down. Upon restart, the new drive locations are noted, and no other reconfigurations are needed.

Spare Drives

Multicore RAID changes the traditional Hot Spare paradigm, because there is no need to define hot spares on the storage system. In fact, with MCx, it is no longer possible to define hot spares. MCx spares failed drives to compatible unused drives.

When a drive fails (excluding the system drives), Multicore RAID:

- Looks for a suitable replacement in the storage system among unused drives
- If a match is found:
    - Add the new drive to the storage pool
    - Start the drive rebuild process

When a system drive (DPE Drives 0-3) fails, Multicore RAID:

- Looks for a suitable replacement (persistent spare) in the storage system among unused drives if user space is present on the system drive
- If a match is found, the persistent spare is used to rebuild the user space from the remaining drives in the RAID group
- The failed system drive must be physically replaced in the system
    - Upon doing so, the VNXe system space is rebuilt to the new system drive
    - User space formerly on the system drive remains on the persistent spare

With MCx, drive failures in the VNXe3200 system immediately look for a suitable spare drive to begin the rebuilding process based off the remaining drives in the RAID parity configuration. When moving drives in your system, as noted in the **Drive Mobility** section, MCx utilizes a 5-minute interval to avoid unnecessary full drive rebuilds, which could last several days. The VNXe3200 easily survives human errors, when an incorrect drive is removed during a routine failed drive replacement procedure. Simply insert the drive back within the 5-minute window to avoid a timely rebuild.

Sparing Rules

Multicore RAID introduces a small, but helpful change to the VNXe Sparing Rules. The sparing rules define the search order and logic for locating the best suitable drive to replace a failed drive. The storage system uses the following criteria to select a suitable drive replacement:

- **Type**—the system looks for all available drives of the same type (SAS, SAS FLASH, SAS FLASH VP, or NL-SAS) as the failed drive.
- **Bus**—the system looks for all available drives on the same backend bus.
- **Size**—the system looks for all available drives of the same capacity (or larger).
- **Enclosure**—the system looks for all available drives in the same enclosure as the failed drive.
- **Rotational Speed**—the system looks for all available drives with the same rotational speed as the failed drive.

The Proactive Sparing technology, which evacuates data off a drive that is about to fail, does not incur the 5-minute delay.

Hot Spare Policy

Multicore RAID introduces the ability to forewarn an administrator about the state of usable spare drives in the system through the use of the Hot Spare Policy. The Hot Spare Policy monitors a number of unused drives (potential spare drives) of a certain type in the system, and posts a warning when that number is low. The VNXe3200 implements a single, non-customizable Hot Spare Policy that keeps 1 hot spare per 30 drives of every type.

Multicore RAID defines a policy for every supported drive type and size, but does not distinguish form factors. For instance, Multicore RAID does not differentiate a 2.5-inch form factor 300GB 15K SAS drive from a 3.5-inch 300GB 15K SAS drive, where sparing is concerned. Both are considered to be the same, and therefore compatible for sparing.

The Hot Spare Policy is specifically defined by drive types and capacity. For example, SAS 300GB drives and SAS 600GB drives each have an independent Hot Spare Policy defined (Figure 15).

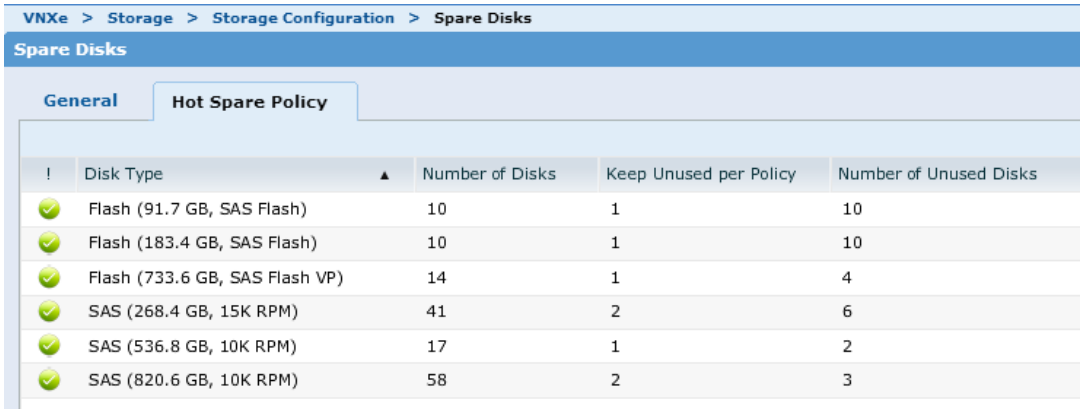


Figure 15 – Hot Spare Policy

When creating a RAID 5 (4+1) Extreme Performance tier in a storage pool with 100GB FLASH drives (Figure 16), Unisphere's Storage Pool wizard would only allow you to add 5 drives, even though there are 10 unused available drives, because 1 drive is reserved by the Hot Spare Policy shown in Figure 15.

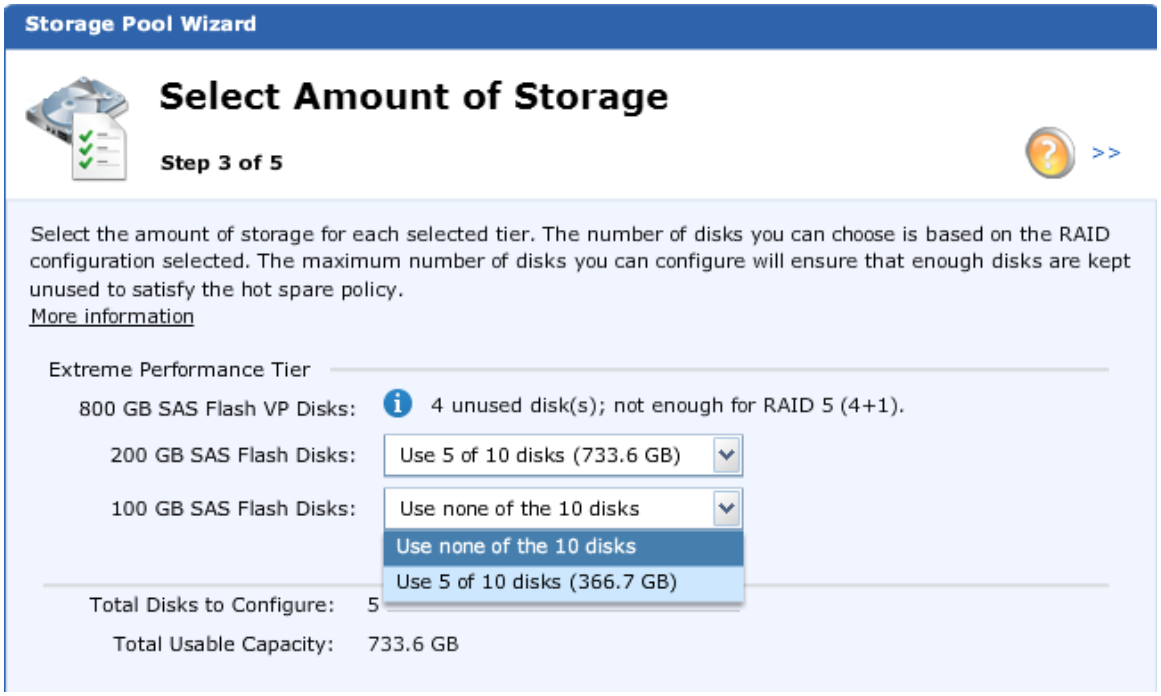


Figure 16 – Create Storage Pool

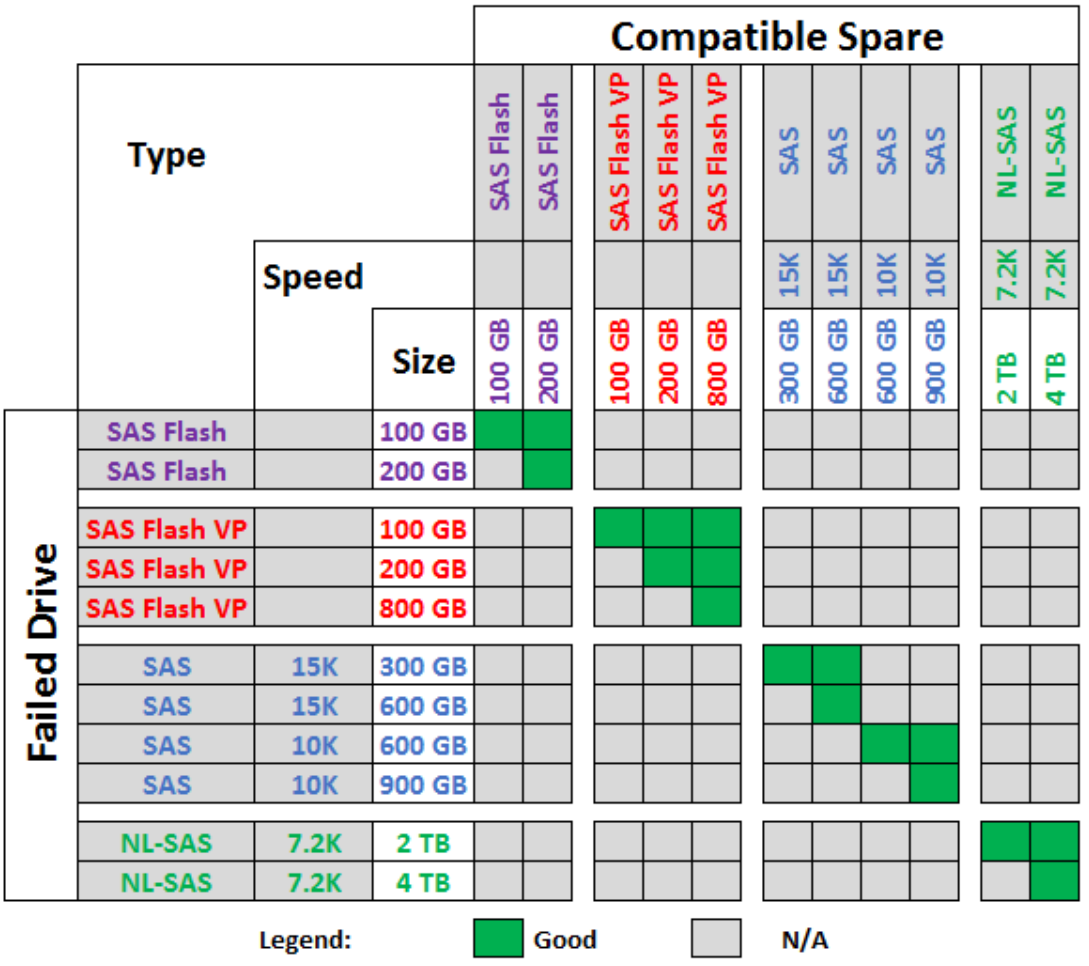
Drive Sparing Matrix

The VNXe3200 supports a broad range of drive types to meet customers' business demands. The wide selection requires the implementation of numerous compatibility rules.

For example, the latest Flash drives have two families: SAS Flash and SAS Flash VP drives. Although these drives seem nearly identical, their underlying technology differs significantly. These two drive types cannot be a spare for each other. However, a 3.5 inch SAS 15K RPM 600GB drive can be a spare drive for a 2.5 inch SAS 10K RPM 300GB drive.

To simplify the choice, and for overall convenience, refer to the following matrix:

VNXe3200 Drive Sparing Matrix



Notes: Sparing does not take into account:
- Form factor (2.5 inch or 3.5 inch)
System drives (DPE Drives 0-3) cannot be used as spare drives

Figure 17 – Drive Sparing Matrix

Storage Pools and Drive Compatibility

Figure 17 shows groups of drives organized by drive type. The compatibility within each group is used for sparing and storage pool RAID configuration creation. The VNXe3200 system does not allow a storage pool's RAID configuration to contain drives of different types. In fact, Unisphere for VNXe only allows the creation of complete RAID sets within a storage pool using the same drive type (taking into account drive size and drive speed).

In Unisphere, the Storage Pool Wizard includes a drop-down box for each drive type:

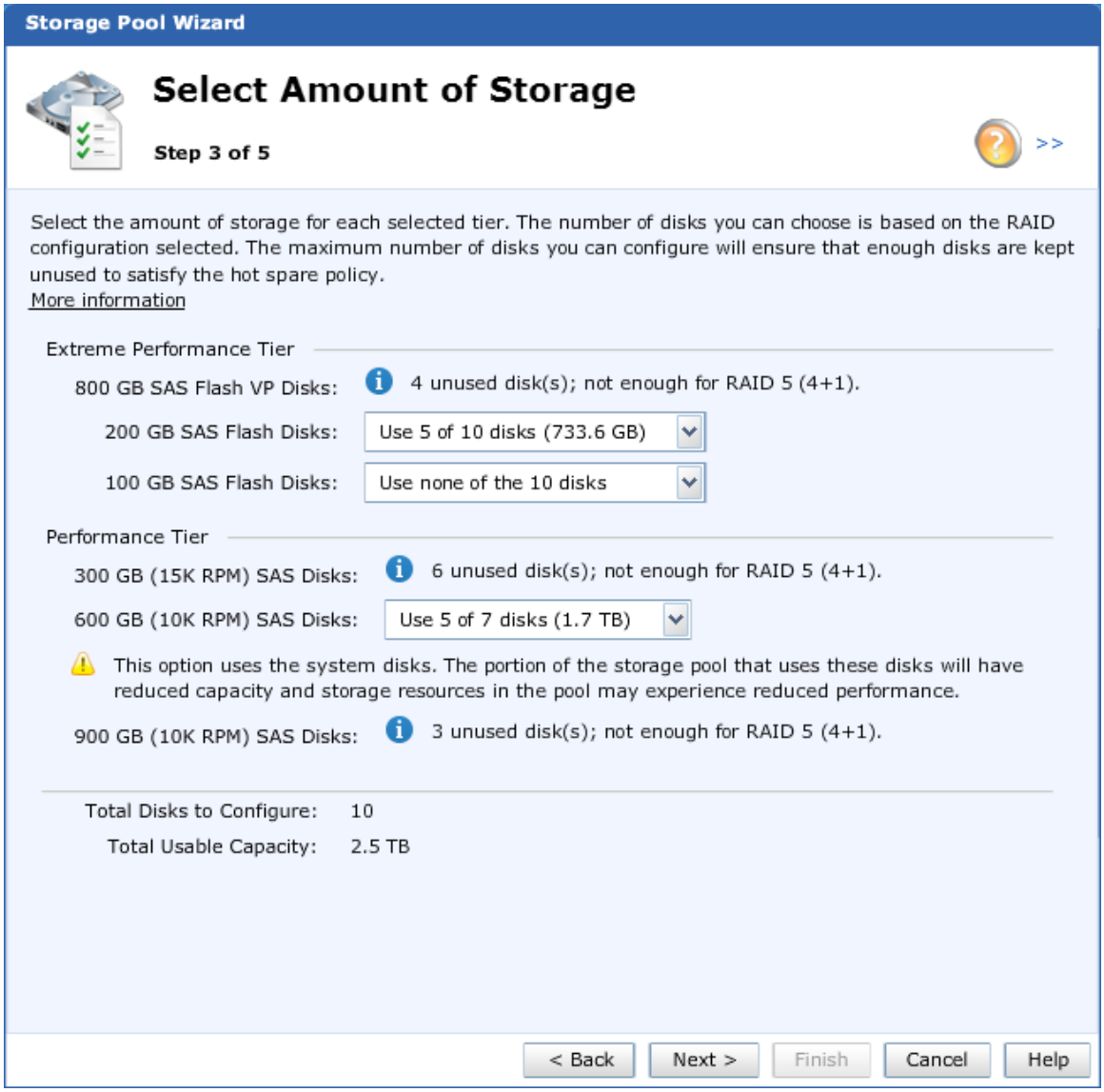


Figure 18 – Extreme Performance Tier

Drive Sizing

In the VNXe3200, Multicore RAID formats drives before they can be used by the system. The formatting allocates end-user usable space, and some Virtual Drive metadata. Placing drives into a storage pool also causes some capacity to be placed aside for the RAID metadata.

RAID 6 Parallel Rebuild

Multicore RAID introduces another technology: Parallel Rebuild. When a storage pool's RAID 6 configuration is missing two members, it rebuilds the spare drives in parallel. The following describes a sample Parallel Rebuild process (**Figure 19**):

1. The first disk in a RAID 6 configuration fails. The storage system immediately locates a suitable spare and begins the rebuild process. The spare is built from the top.
2. After a spare is 30 percent built, a second drive fails. The storage system immediately locates a suitable spare and begins the rebuild process.
3. In this case, the rebuild process begins at the same block that is rebuilt on the first spare, starting from the 30 percent mark. Both spares continue to rebuild in parallel.
4. When the first spare is 100 percent done, it is a full member of the RAID configuration. The second spare wraps around and starts rebuilding from the top, until it reaches the 30 percent mark, where the rebuild process started.

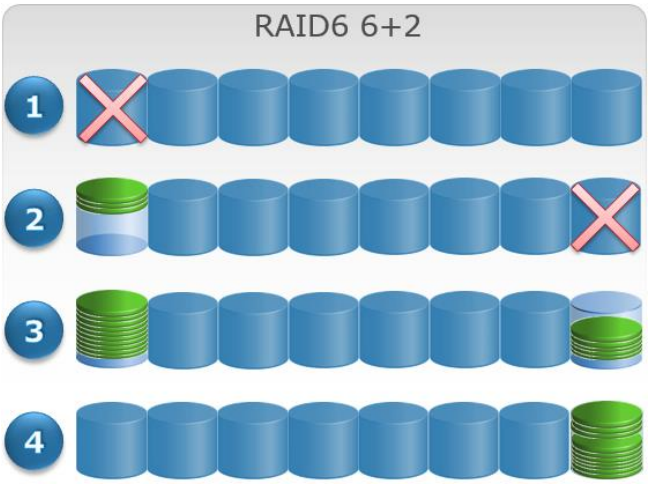


Figure 19 – RAID 6 Parallel Rebuild

New Drives

All unrecognized drives with serial numbers that have not been previously seen by the storage system are zeroed. All drives are zeroed in parallel. Zeroing begins the moment the system discovers the new drive or upon first power on of the system. The drives are available while they are being zeroed.

Rebuild Logging

When an actively used drive goes offline (for example, it fails, is pulled out of the slot, or is temporarily unresponsive for some other reason), Multicore RAID enables a RAID protection mechanism called Rebuild Logging. The storage pool's RAID configuration itself is said to be in a degraded state when one of its drives is missing. This is true for all RAID types[7].

Rebuild logging marks every data block that should have been updated on a missing RAID member. The marking is stored as a block map as part of the RAID group metadata. RAID metadata is stored within the storage pool drives themselves. Think of RAID metadata as a small hidden LUN at the bottom range of the storage pool RAID configuration.

For the following RAID 5 4+1example, refer to **Figure 20**.

1. A drive in the RAID 5 4+1 set is removed from the VNXe3200 system or briefly goes offline.
2. The parity group keeps a log of all the areas that are updated while the drive is missing from the RAID set.
3. If the drive was removed and reinserted back into the VNXe3200 system within the 5-minute window, only the changed areas are rebuilt based on the information in the log.
4. The RAID set is now healthy and fully operational again.

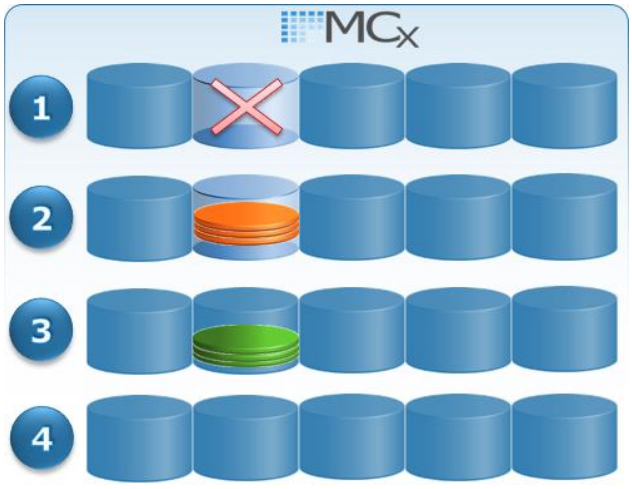


Figure 20 – Rebuild Logging

The log is a bitmap created during the storage pool RAID configuration creation as part of the metadata. This log defines the size of the RAID overhead. Therefore, it

[7] VNXe3200 supported RAID types are: RAID 6, RAID 5, and RAID1/0.

cannot run out of space. The log is persistent; a rebuild process interrupted by a reboot continues after the power is restored.

Rebuild logging is active for the entire duration that a storage pool is in a degraded mode. If a storage system does not invoke a spare (for example, when unused compatible drives are not present in the system), rebuild logging continues to run until the storage pool is brought offline or a suitable spare drive is found.

Rebuild logging enables the VNXe3200 to avoid full RAID configuration rebuilds. With this functionality, the VNXe3200 can reuse good data from a drive that was temporarily inactive, significantly reducing the need to rebuild entire drives.

Write Journaling

Data written onto a VNXe3200 drive consists of two parts: the User Data and the Cyclic Redundancy Check (CRC)[8]. The checksum provides basic protection against media errors. During the read, the VNXe3200 calculates a new CRC and compares it with the one on the drive. If the two match, the data is good. If they do not match, the block is marked as bad and a checksum error condition is logged.

An internal VNXe process called Background Verify examines the health of stored data on the drive. It reads checksums on existing data and compares them to calculated values. When a mismatch is found, Background Verify attempts to fix it using RAID protection. Another process, called Sniffer, checks the physical state of the drive. It marks a bad media block as media error.

The number of mismatches and media errors on every drive is recorded and counted. When enough errors are recorded, the drive is declared unhealthy, and a Proactive Sparing process starts.

Even though Background Verify and Sniffer can find errors, they cannot recover data on degraded parity RAID groups. Media errors occur because of natural causes, such as a bad magnetic condition of the plate, Flash data decay, excessive shaking of the spinning drive, and so on.

A new Multicore RAID technology called Write Journaling writes a journal entry onto the same drive before proceeding with an actual write (**Figure 21**). If the journal write is committed without issues, the Multicore RAID system proceeds to commit the write to its intended address on the drive. After the write is successfully committed to the drive, the journal entry is invalidated and the write is marked as clean in Multicore Cache.

8 Cyclic Redundancy Check, "Applications of Matrix Algebra to Network Theory," Circuit Theory, IRE Transactions on, vol.6, no.5, pp.127, 137, May 1959.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1086603&isnumber=23613

Figure 21 – Write Journaling

Uncommitted dirty Multicore Cache pages are protected by the vaulting mechanism (discussed in the **Multicore Cache Vaulting** section). After recovering a RAID group from any fault condition (for example, powering on after an unscheduled power off, recovering after several drives go offline, and so on), the journal area is examined for active journal entries. If any are found, they are played back to proper locations.

As with rebuild logging, Write journaling is active only when a RAID group is in a degraded state—the two technologies coexist. Write journaling writes the journal to the RAID group metadata area within the destination drive (Error! Reference source ot found.). The journal contains the actual write plus metadata. Journal space is limited and managed on the circular buffer basis. When journal space is full, incoming writes are delayed until first entries are invalidated. Write journaling comes with some performance degradation: every drive write is now doubled.

Background Processes

Multicore RAID has several background processes that ensure data integrity for your VNXe3200 storage system.

Drive Zeroing

MCx changes how the VNXe3200 systems treat new drives. Multicore RAID always zeroes new and unknown drives inserted into the system before using them. If a drive is moved between compatible[9] storage systems, it is always zeroed first. Drive zeroing is a background operation, but it may affect performance. Drive zeroing cannot be disabled.

Note: New storage systems may appear to have initial slower performance due to drive zeroing.

MCx allows the new disks to be used while they are being zeroed. They can be placed into a storage pool and additionally, storage resources can be created and data can be written.

Multicore RAID keeps track of the zeroed regions of the drives. When a Host I/O is requested (read or write), Multicore RAID first checks the zeroing map. If the Host I/O is addressing a region that was already zeroed, the I/O is allowed to proceed. If

[9] Please check storage system and drive compatibility with the EMC E-LAB Interoperability Navigator.

the Host I/O is addressing a region that has not been zeroed, Multicore RAID first zeroes that region, and then allows the I/O to proceed.

Note: VNXe3200 drive zeroing technology does not comply with secure erasure regulations or standards. Use other approved data cleanup means if compliance is required.

Sniffer

Sniff Verify, or Sniffer, is an internal process that detects and corrects potential disk media errors before they become unrecoverable. The Sniffer process operates on the entire disk region.

Background Verify

Background Verify is an internal process that inspects and validates the data and parity of bound LUNs in order to detect and correct RAID errors. The Background Verify process will lessen its impact on the RAID group to allow it the time needed to process I/O when the RAID group is under high load.

NOTE: Background Verify only works for private Pool LUNs.

Background Verify is started for a few different reasons:

- User manually initiates the operation.
- System initiates the operation automatically for incomplete write error situations.
- System initiates the operation automatically when background zeroing on a private Pool LUN is complete.

Proactive Copy

The VNXe3200 system monitors drive health by counting the number of errors found. The errors are discovered by the Sniffer or Background Verify processes, or during regular host reads. When the number of errors reaches 60, the drive is declared unhealthy and invalidated. Multicore RAID will begin the sparing process to a compatible spare drive. Drives do not increment their error count forever. A drive's error count is decremented by one after 18,000 healthy I/Os to the drive.

System Drives

MCx changes VNXe3200 system drives requirements. The VNXe system space occupies the first four drives of the Disk Processor Enclosure (DPE). However, with MCx, the size of the system area increases to almost 80GB per drive.

The VNXe system space includes:

- The VNXe configuration data LUN.
- Utility mirror partition for SPA (0, 2) and SPB (1, 3) that holds the installation image, bootflash backup image, and logs/dumps. POST validates bootflash image on mSATA at startup and will reimage from backend copy if corrupt.

- Control LUNs, triple mirrored on disks 0,1,3 – containing NASDB and logs.

Everything in the system space is hidden from user view. The system drives (DPE drives 0-3) can be used in storage pools; however, the RAID group set they are in has reduced capacity. For example, if a VNXe3200 system has 600GB SAS system drives, the usable space on these drives is ~459GB, whereas the usable capacity of the other 600GB SAS drives in the system is ~536GB usable capacity.

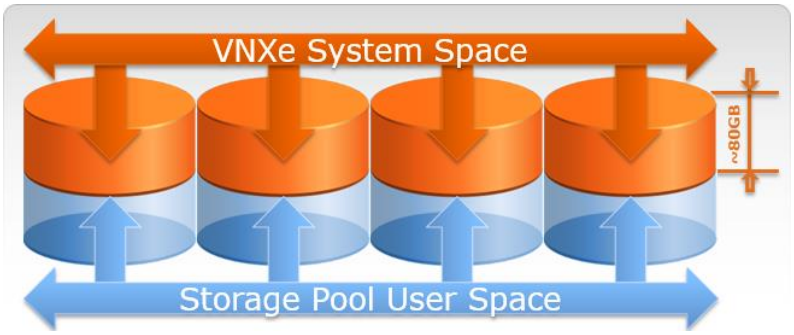


Figure 22 – VNXe3200 System Drive Space

The system space does not require a spare because the VNXe3200 system uses special internal protection for the data.

Note: A failed system drive must be physically replaced.

The system drives with end-user usable space act like regular drives with adjusted usable space. They can be placed into storage pools; however, a warning will be shown indicating a reduction in performance on these drives. When a system drive fails, the user space data is spared onto a suitable spare drive, as noted in the **Spare Drives** section.

If a system drive is moved to another slot, the VNXe3200 system:

1. Marks the system drive as removed.
2. Formats and zeroes the former system drive in the new slot.
    a. All data (including the private vault system and user spaces) on the former system drive is removed (zeroed).
    b. If the moved system drive was a member of a RAID group, the user space portion is spared with normal RAID rebuild procedures after a 5-minute interval.

If a data drive is moved to a system slot (0_0_0 through 0_0_3), the behavior is almost the same, with exceptions noted in italics. The VNXe3200 system:

    a. Marks the moved drive as removed and *starts a 5-minute counter* toward sparing the moved drive to a suitable replacement.
2. Formats and zeroes the drive in the system slot.
    a. *Reserves ~80GB system space*.
    b. Leaves the remaining space unused (if any).

Multicore RAID Summary Examples

For the purpose of the following scenarios, review the 4+1 RAID 5 example in Figure 23. When Member 4 fails, the RAID group becomes degraded, and both rebuild logging and write journaling start. A 5-minute timer activates, which delays the sparing process. The performance of the RAID group slows down due to write journaling overhead.

Figure 23 – Rebuild Logging and Write Journaling

A host writes 8KB to LUN1, address 0x400. For the purposes of this example, 0x400 is located on member disk 0 of this RAID group. The write is committed in Multicore Cache and marked as dirty. At some point, Multicore Cache will flush this write to the drive. This prompts a read from Members 1 and 3 to calculate parity. Now, parity and host I/O writes are written to the RAID group —original host I/O to member 0 and parity to RAID Member 2.

Since the RAID group is degraded, journal entries are first written onto both disks. After the journal entries are committed, the two writes proceed onto their normal locations. After they are successfully committed, rebuild logging marks the blocks as updated, and write journaling invalidates the journals.

Scenario A: The lost drive comes back online

When a failed drive comes back online (for example, it was moved to another slot), the rebuild log is examined, and the blocks marked as changed are rebuilt on Member 4. Write journaling stops, rebuild logging stops, and the RAID group mode changes back to normal.

Scenario B: Another drive fails

If one more disk fails (for example, Member 0), the storage pool becomes faulted. A degraded RAID 5 group cannot survive two disk failures. A RAID 6 group, however, remains in degraded mode, unless three disk members fail. When moving live drives within the VNXe3200 system, be aware of the increased failure risk.

Figure 24 – Another Drive Fails

To answer whether there will be data loss if a second drive fails, consider the following scenarios. The answer depends on the cause of the failure. For the following scenarios, please refer to **Figure 24**.

B1. Member 0 drive is removed

   a. It will come back at some point, with all data intact.

B2. Member 0 drive is completely dead

   a. It will never come back.

B3. DAE is shutdown

   a. The DAE will either be powered back up, or

   b. The drives on it will be moved to other slots.

For each of these scenarios, assume that the rest of the storage system is operating normally.

Scenario B1

The moment Member 0 drive is removed, the storage pool's state changed to faulted, and it stops accepting I/Os.

The data written onto other drives remains intact. The data written onto Member drive 0 is protected by write journaling; even though it may have a write hole. The data that has not been flushed from Multicore Cache is still in Cache—the flush ACK has not been returned, and data continues to be marked as dirty. Multicore Cache will post a warning that the dirty data cannot be saved due to a drive failure. All I/O activity to the RAID group is suspended. The hosts will receive I/O error replies.

When and if Member 0 comes back, write journaling is examined, and the valid log entries are played back. The RAID group is examined for consistency and if no

issues are found, the RAID group is brought back online. Rebuild logging and write journaling restart, and the dirty cache is flushed.

In this sub scenario, data unavailability occurred until Member 0 was returned, but there was no data loss.

Scenario B2

If Member 0 is faulted and unrecoverable, the data from it is no longer available. There is no way to recover this data. This is a full RAID group data loss.

Scenario B3

This scenario is similar to scenario B1. All drives become unavailable at once, so write journal and rebuild log have consistent entries. Upon DAE power up, or moving the drives to new locations, the storage system can restart this RAID group.

Scenario C: 5-minute interval expires

After the 5-minute interval expires, the storage system looks for a suitable spare drive. When one is found, the RAID group rebuild process begins. LUNs are rebuilt sequentially, from the top of the drive to the bottom. Rebuild logging and write journaling processes continue until 100 percent of the drive is rebuilt.

If Member 3 drive comes back online in the middle of the rebuild process, it is considered a new drive and is zeroed. This is because the old Member 3 drive is no longer a part of the RAID group, and its content is no longer relevant. VNXe does not interrupt an ongoing rebuild operation of the RAID group.

Summary

MCx is one of the most advanced storage system stacks available. As described in this paper, it offers substantial innovations in hyper-efficient horizontal multicore system scaling.

In addition to increased platform efficiency, the multicore-optimized stack also delivers rock-solid data protection and media management. With MCx, Multicore FAST Cache sits in the correct position in the storage hierarchy, enabling DRAM accelerated writes while delivering up to 400GB of data cache.

MCx drives the VNXe3200 storage experience to new heights and ensures smooth, efficient, and reliable performance.

References

For additional information regarding topics mentioned in this document, please refer to the following documents, available on EMC Online Support:

EMC Unisphere for VNXe3200: Next Generation Storage Management white paper

EMC VNXe3200 FAST Suite white paper

Introduction to the VNXe3200 white paper