White Paper

EMC VNX2 MCx

Multicore Everything

Abstract

This white paper explains the EMC® VNX™ MCx™ technology, and its main components: Multicore RAID, Multicore Cache, and Multicore FAST Cache. Together, they offer the VNX2 series features such as Multicore Scalability, Symmetric Active/Active, Adaptive Cache, Permanent Spares, and many others.

May 2016

Part Number H12090.8

Table of Contents

CLI Commands

Figures

Introduction

EMC's midrange VNX systems have emerged as the de-facto standard for virtual server deployments. Typical deployment sizes vary from a few dozen to several thousand virtual machines. Keeping virtual machine seat cost to a minimum is common to all deployments.

A modern storage system must deliver higher IOPS than ever before, a need driven by modern multicore/multisocket server architectures. Until recently, midrange storage arrays served 20-40 CPU cores: today's arrays must serve 200-400 CPU cores or more. Moore's Law—higher granularity silicon geometry and 3D chip design for the future servers—promises to sustain that trend. An array 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 also continues to climb. A few years ago, a typical midrange array only needed to store 10-20TB of data. Today, midrange arrays' 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 re-build times are now prohibitively high.

Increased capacity places new demands on the array cache. As system capacities grow, the system cache must follow. 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. Larger VNX systems pose a problem: a 1500 disk-slot system can house an aggregate capacity of 6PBs, 0.1 percent of which is 6TB. Currently there are no dual controller designs capable to deliver this much DRAM. In addition, 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 NAND 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 VNX2 systems with MCx delivers unprecedented performance and platform efficiency. This paper discusses some of the key innovations of MCx.

Audience

This paper is intended for anyone interested in learning about the MCx and how this advanced I/O kernel changes what is possible in a midrange storage system like VNX.

Basic understanding of storage and SAN principles is expected.

MCx Goals

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

- Scale on multi-processor architectures
- Deliver Active/Active capability
- Improve performance and scalability
- Optimize memory efficiency

CPU Scaling

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 microarchitecture offers (8-cores in Sandy Bridge, 10-cores in Ivy Bridge today as well as higher core counts in the future). 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. Intel's Haswell microarchitecture fully demonstrates this trend [5].

FLARE®

To explain the innovation of MCx, you must first understand the FLARE Operating Environment (OE) architecture. FLARE, which runs on the CLARiiON CX and original EMC VNX Series of arrays (VNX1), was designed with a single threaded paradigm. Every thread runs on a dedicated CPU core. Some FLARE functions are bound to the first core, also known as core 0. The incoming I/O processes on core 0, and is then parceled out to the other cores. As this core has the most CPU intensive functions, it often becomes the gatekeeper to the other processors.

When FLARE runs on a relatively modest number of cores—two, four, or even six—the single-threaded paradigm works reasonably well. Figure 1 illustrates the core utilization in FLARE:

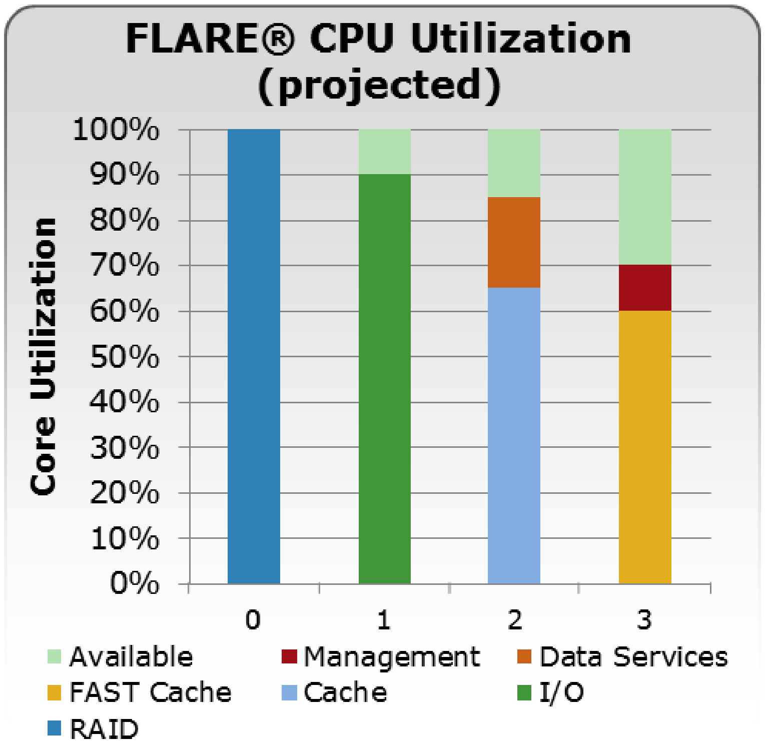


Figure 1. FLARE CPU with four cores

There are more threads than available CPU cores, so no cores are wasted. However, with a growing number of CPU cores, the potential bottleneck of core zero affects the overall core budget. VNX engineering experimented with FLARE and larger CPU-core counts. As expected, core 0 stayed 100 percent busy, core 1 at 90 percent, core 2 at 65 percent, and so on. Adding more CPU-cores did not provide additional benefit as cores 6 through 15 could not be kept busy at all, as shown in Figure 2. A new multi-core optimization approach was needed.

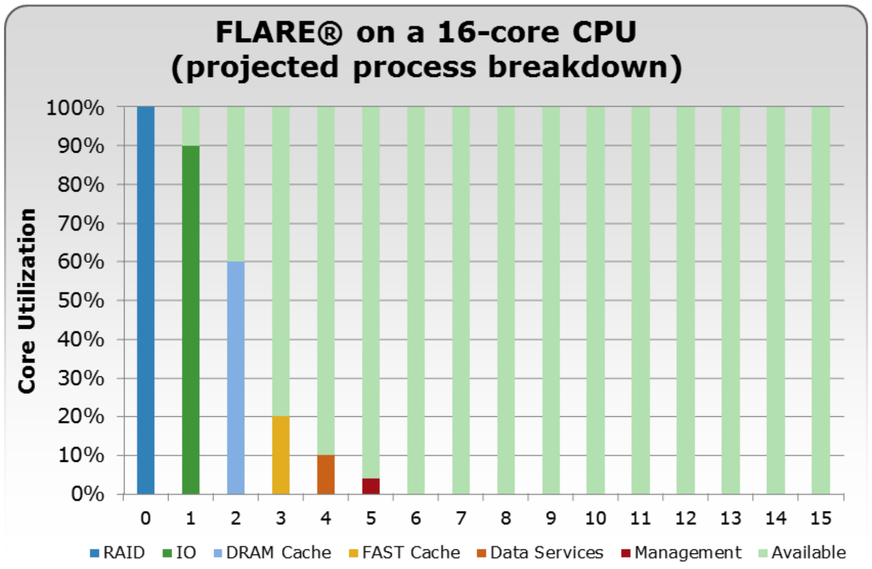


Figure 2. FLARE on a 16-core CPU

MCx

MCx changes the way the VNX2 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.

Core Affinity

MCx has a concept of a *preferred core*. Every front-end and back-end 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 back-end, every core has access to every drive, and servicing requests to the drives do not require swapping with another CPU. When a preferred core is busy, an alternate core is used.

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

MCx leverages the CPU processing ability more effectively than FLARE, directly increasing performance. MCx also enables scaling performance to much higher levels than FLARE.

Figure 3 shows the result of a VNX2 array (running MCx) subjected to a workload as part of the standard VNX performance testing. Note that only eight front-end ports were used as part of the testing.

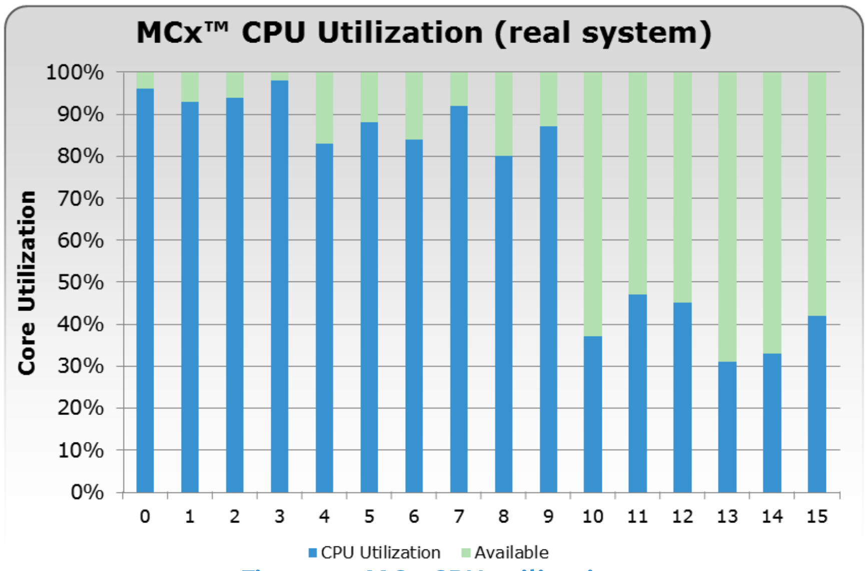


Figure 3. MCx CPU utilization

Higher-numbered cores that are less used than the rest, indicates that the workload is not equally balanced across all system front-end ports. Amdahl's law stipulates that it is not always cost effective to keep all cores busy at the same level—the graphic above shows a healthy system. The array is busy, but does not hit its performance ceiling.

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, as shown in Figure 4.

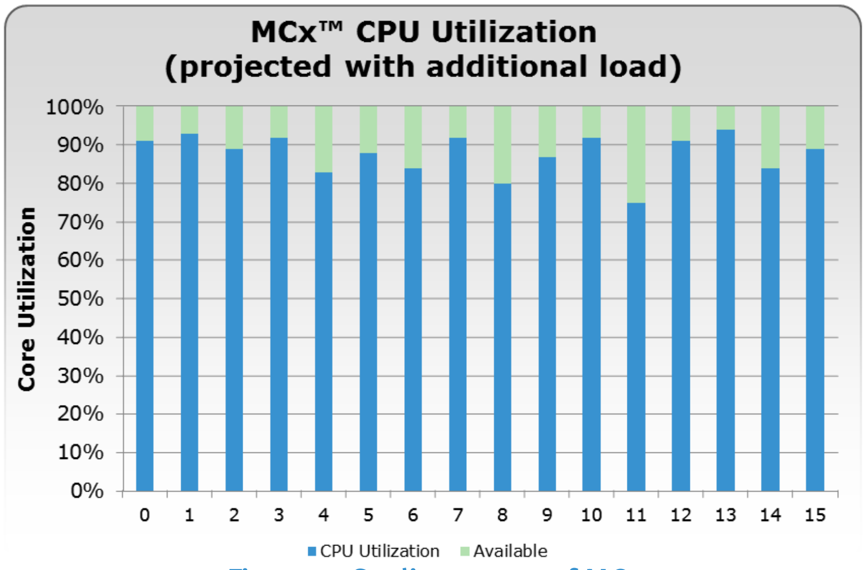


Figure 4. Scaling power of MCx

Stack Components

Historically, the VNX Block Operating Environment pre-dated multicore technology and was not designed to leverage CPU scaling dynamically. FLARE was born when the RAID definitions were being developed, and became the midrange storage engine for physical drive access, RAID, and caching.

Over time, additional capabilities and features supplemented FLARE: Pools, Compression, Auto-Tiering, Snapshots, and so on. Collectively, this document refers to these features as *Data Services*.

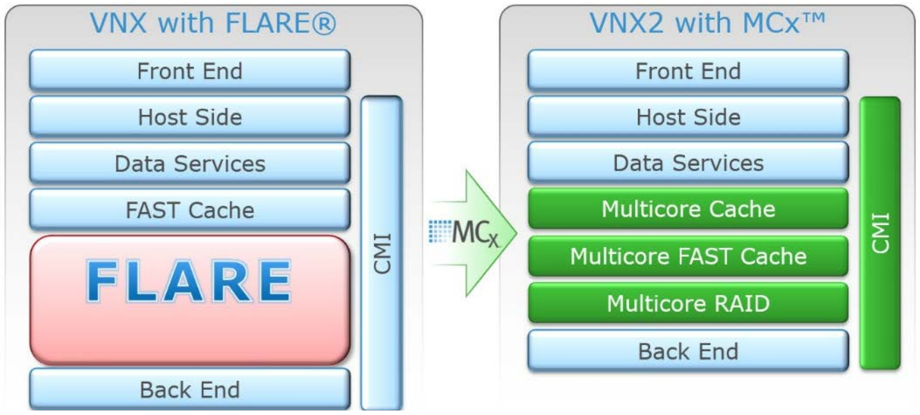


Figure 5. VNX stack components

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 with a reinvention of core components:

- Multicore RAID
- Multicore Cache
- Multicore FAST Cache
- Symmetric Active/Active
- Multi-Path Communication Manager Interface

A significant architectural change is the stack order itself. With the FLARE stack, all I/O goes through the FAST Cache layer's memory map before it reaches the FLARE layer where DRAM Cache lives. This behavior is largely an artifact of adding FAST Cache to an already mature FLARE stack. As a result, there is a delay in sending a committed write acknowledgement back to the host, limiting write performance.

With MCx, all writes go directly to the DRAM Cache and considerably reduces host latency. As a result, host writes are mirrored and protected directly from the DRAM cache layer and increases performance. See the Symmetric Active/Active section for more details on the effects of the stack order.

MCx introduces new Symmetric Active/Active technology, which allows both SPs to write data directly to the LUN and bypass the need to send updates to the primary SP, as FLARE requires.

Multicore Cache

The VNX2 array's *Multicore Cache*, also known as *SP Cache*, is an MCx software component that optimizes a VNX2 storage processor's DRAM to increase host write and read performance.

Just like Multicore RAID, Multicore Cache was designed to effectively scale across multiple CPU cores. In FLARE, cache is a part of the main FLARE driver. In MCx, Multicore Cache is a separate component within the MCx stack. The cache ties all other components together. Multicore Cache provides write caching for all VNX software layers (Refer to Figure 5).

Multicore Cache features:

- Multicore CPU scaling
- Symmetric Active/Active support
- Adaptive and automatic performance self-tuning

Adaptive Cache

Before reviewing Multicore Cache improvements, it is useful to review how the FLARE cache works. FLARE divides its cache into three regions: read cache, write cache, and peer SP mirror write cache. This division does not include OS overhead.

As illustrated in Figure 6, the primary function of the read cache is prefetching. *Read Misses* result in data being loaded into the read cache (1). Host read requests can be satisfied from either read or write cache, if needed data is available when the request comes in; or it can be satisfied from read cache (2) if the data needs to be loaded from disk.

The write cache is similar, except that every incoming write I/O (3), is marked "dirty" and then mirrored to the peer SP cache (4). The host can overwrite dirty page while it is still in cache. This is known as a *Write Hit.* Write cache uses a Least Recently Used rule (LRU) to de-stage the oldest unused cache page to disk. The de-stage action is called *Flushing*. At a certain point, due to the LRU rule, the cache page is written out to disk (5), and both copies are discarded from each SP (6).

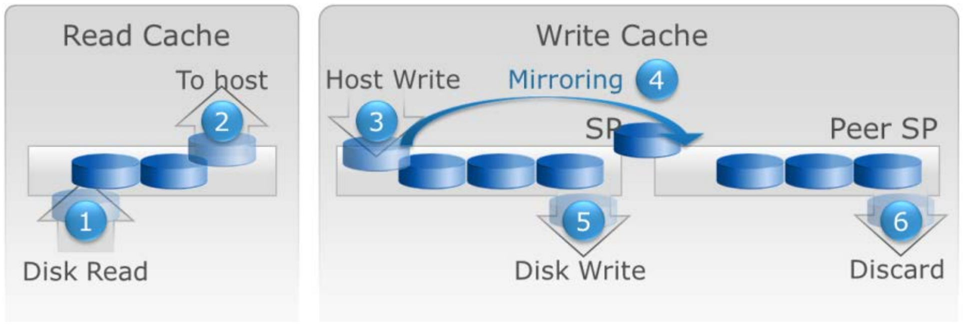


Figure 6. FLARE cache

Unlike FLARE Cache, Multicore Cache space is not split between read and write functions. Instead, the cache is shared for writes (1) and reads (2) as shown in Figure 7. Furthermore, instead of flushing a dirty cache page to disk, the page is copied, called cleaning, to disk (3) and therefore it remains in memory and is not discarded. This way the page can be hit again by the host (4) and (5), improving performance. Data is ultimately expelled from cache (7), which frees the page for other workloads.

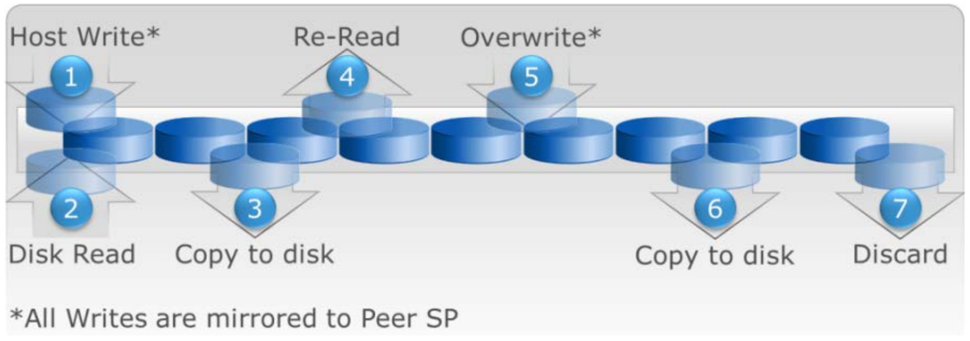


Figure 7. Adaptive cache

Mirroring is important—VNX guarantees data *durability* the moment the write I/O hits the cache. A dirty page is always copied to the peer SP cache. If one SP fails, the remaining SP has a pristine 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 that are owned by the peer. Page ownership changes if one of the SPs becomes unavailable (for example, if it panics, crashes, reboots, and so on.). Each SP is responsible for cleaning its own dirty pages to the back-end, clearing the dirty flag in both caches.

Note: Multicore Cache always mirrors every committed write I/O to the peer SP cache, 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 (similar to FAST VP slice temperature), 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, if the array has been up for a year, what has happened in the last hour is much more important than what happened last month.

Pools also fully benefit from all Multicore Cache innovations. 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. That is fundamental to the design of Pools. Multicore Cache monitors and manages traffic to the underlying RAID groups in Pools, just as with user-managed RAID groups.

Flushing

FLARE has a condition called *Forced Flushing*. It occurs when the percent count of dirty cache pages crosses over the high watermark and reaches 100 percent. At that point, the cache forcefully flushes unsaved (dirty) data to disk and suspends all host I/O. Forced flushing continues until the percent count of dirty pages recedes below the low watermark.

Forced flushing affects the entire array and all workloads served by the array. It significantly increases the host response time until the number of cache dirty pages falls below the low watermark. The Storage Processor gives priority to writing dirty data to disk, rather than allocating new pages for incoming Host I/O. The idea of high and low watermark functionality began as a mechanism to avoid forced flushing. The lower the high watermark, the larger the reserved buffer in the cache, and the smaller chance that forced flushing will occur.

Dynamic Watermarks

Multicore Cache changes the cache-cleaning model. There are no pre-set dirty cache page watermarks. Instead of keeping track of just the dirty cache pages, Multicore Cache also monitors the LBA corresponding to these pages reference, as well as the demand that hosts put on Multicore Cache and the underlying a 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. As described earlier, excessive flushing does not improve performance, on the contrary, it debilitates all system workloads.

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 flushes.

The evaluation allows Multicore Cache to react dynamically to workload changes: workloads exhibiting signs of possibly overwhelming the cache may be subjected to write throttling; temporary short bursts may be fully buffered by cache; small block random writes may be coalesced for improved disk de-staging; and so on.

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.

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
- The rate of successful page flushes to the RAID group
- The number of available/free cache pages
- The effectiveness of the workload coalescing

To determine the correct time for a cache dirty page to be cleaned 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 cleaned. If the cache page age is larger than its Pre-Cleaning Age, the Multicore Cache may 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 protects its resources from being over consumed by a single workload addressing an overwhelmed RAID group.

Write Throttling

The ability to auto-adjust incoming and outgoing I/O per RAID group is the new *Write Throttling* feature of the Multicore Cache. Write Throttling buys the cache the time needed to adjust 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 time needed to process an increased flushing load by delaying host I/O acknowledgements, thus slowing incoming write I/Os. Figure 8 shows this process.

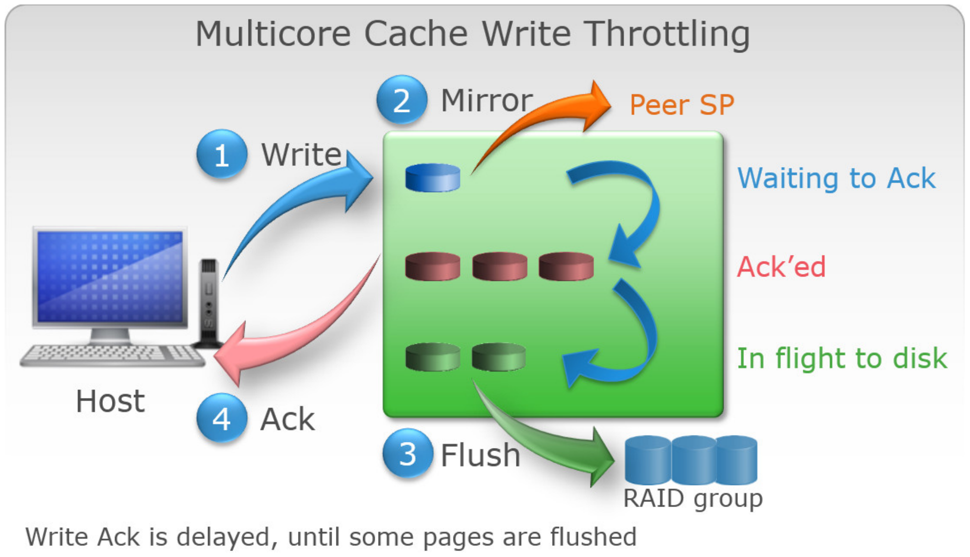


Figure 8. Multicore Cache write throttling

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 pursues the following goals:

- To protect the system resources from being monopolized by a single workload
- To 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. 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, Write Throttling stops.

As said in the previous section, 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 may increase. If the activity grows further, write throttling may be used to decrease the value further.

Write-Through

Multicore Cache read cache is always enabled. Multicore Cache 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. Write I/Os are not acknowledged until successfully written to disk. There is also no mirroring with the peer SP. Only when the flush completes successfully is the I/O acknowledgement sent to the host.

Write Coalescing

For spinning drives, Multicore Cache offers substantial performance gains when writing IOs to adjacent LBAs. For that purpose, Multicore cache is using *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 array reports physical page size in the CLI as `Cache page size`:

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0 cache
–sp -info

SP Read Cache State:  Enabled
SP Write Cache State:  Enabled
Cache Page size (KB):  8
Read Hit Ratio:  67
Write Hit Ratio:  15
Dirty Cache Pages (MB):  4
SPA Read Cache State:  Enabled
SPB Read Cache State:  Enabled
SPA Write Cache State:  Enabled
SPB Write Cache State:  Enabled
SPS Test Day:  Saturday
SPS Test Time:  21:00
SPA Physical Memory Size (MB):  65536
SPB Physical Memory Size (MB):  65536
```

CLI 1. Show SP cache information

The second page type is known as *Logical Page*, and its size is 64KB. One logical page contains from one to eight physical pages. Multicore Cache uses logical pages within its algorithms, as shown in Figure 9.

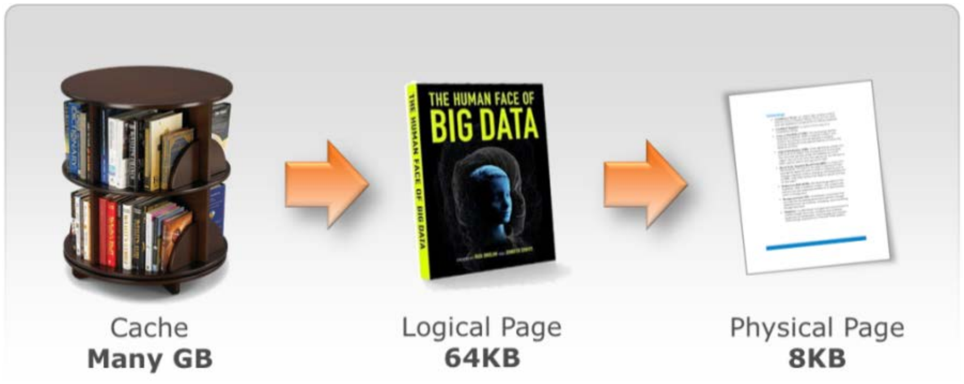


Figure 9. Multicore Cache logical vs. physical page size

Knowing the different page types is important when working with Unisphere Analyzer. Multicore Cache keeps a statistic with the number of allocated logical pages, and a number of dirty logical pages. Typically, only some physical pages are dirty in a set of logical pages. To get the SP Cache Dirty Page metric (displayed by Unisphere Analyzer), Multicore Cache makes an approximation of that ratio based on averaging the write I/O load. Every Storage Processor makes its own calculation, and as a result, the Analyzer graph may 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 transfer rate for a drive is 80MB/s and the average access time for 8KB is 3ms, then 240KB could be accessed and transferred in 6ms, whereas it could take 90ms in 8K units. 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 read-related criteria as follows:

- 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 help Multicore Cache to determine whether prefetching would have been beneficial had it been executed at the time of the prior page, as shown in Figure 10.

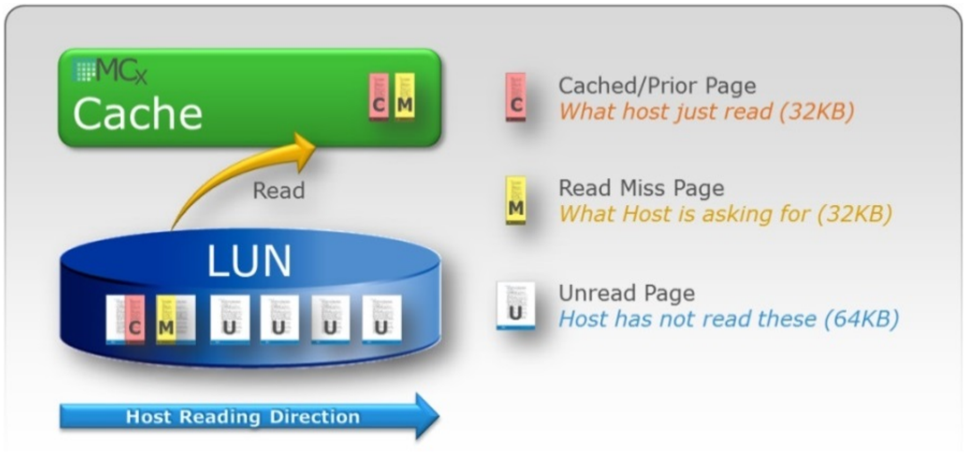


Figure 10. Read miss

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), may 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, q-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, as shown in Figure 11. If a host requests regions that cross logical page boundaries, both pages are loaded (not depicted).

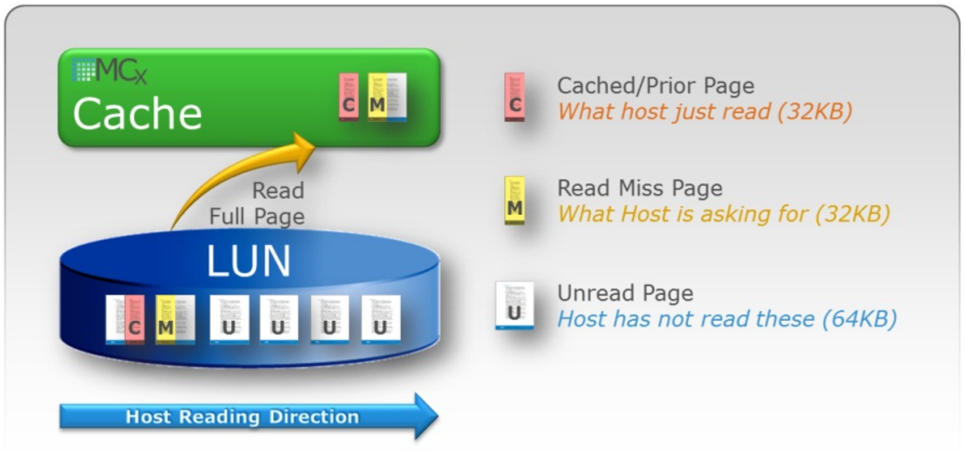


Figure 11. Full page prefetch

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[1] size 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.

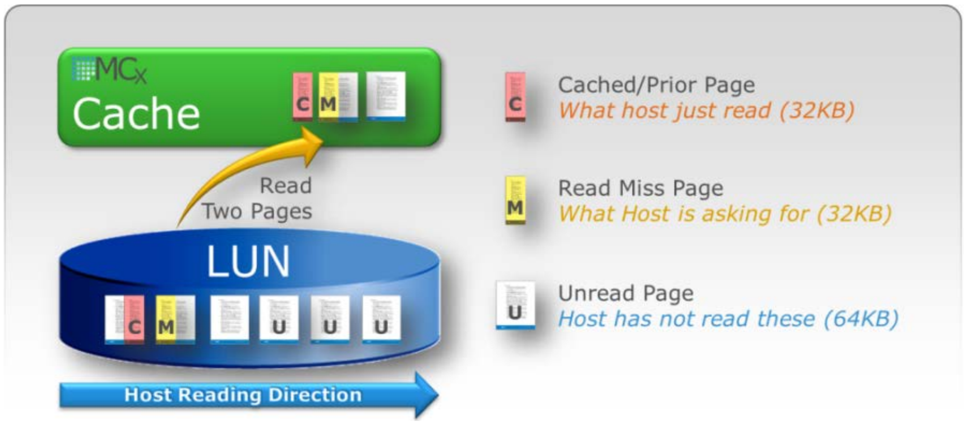


Figure 12. Read more prefetch

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 are used to determine how aggressive Read More will be during the next Read Miss.

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 LUN.

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

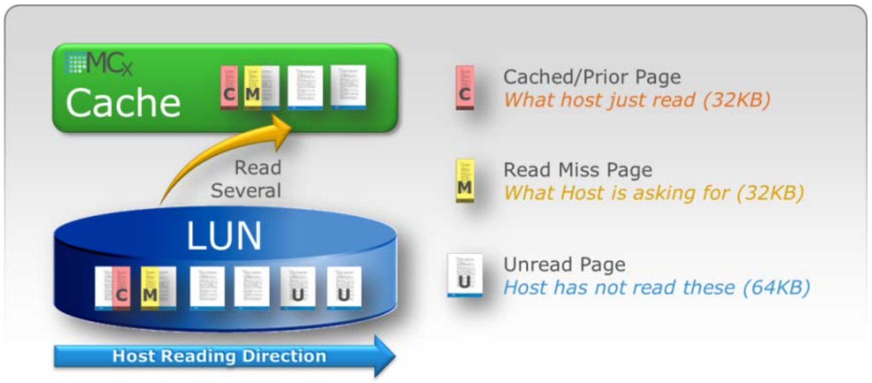


Figure 13. Read ahead prefetch

Typically, read ahead prefetching occurs on larger sequential I/Os—for example, ones with a 256KB I/O size. Please note that 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 VNX2 protects Multicore Cache dirty pages with backup batteries. Different models use different battery configurations, but their functionality is consistent across the VNX product line. In the event of an AC[2] power fault, the batteries will keep the SPs and the first DAE (where the system drives are located) powered long enough to be able to de-stage dirty cache to the vault.

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

Refer to Failure handling section for a full list of hardware faults.

Disabled State

If Multicore Cache determines that it needs to enter the *Disabled* state, it flushes all dirty pages to disk. The flushes are aggressive, but not at the expense of host I/O, Multicore RAID SCSI priority enhanced queuing rules still apply (see SCSI Priority section for more details).

After the flush 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.

Dump to the Vault

If Multicore Cache determines that a vault dump is necessary, it immediately enters a quiescing 5-second interval that gives hosts time to complete any in-process IO (for

[2] AC is used as a generic example; any outside power fault will behave the same way.

example, a 256KB I/O is actively being sent). If possible, this IO is mirrored to the peer SP. While waiting, no new I/O is accepted by the cache. Multicore Cache also sends a message to stop to all background processes (such as drive rebuilds, drive and LUN zeroing, background verifies, sniffing, and so on).

Then Multicore Cache flushes all dirty pages to the vault. After the dump is complete, the SP continues the orderly shutdown. When Multicore Cache starts a vault dump, the "Do Not Remove" LEDs on both SPs are lit, providing a visual indication that the SP should not be removed. The "Do Not Remove" LEDs are turned off when the vault dump completes.

A full system reboot is required to recover the array after a vault dump.

AC Power Glitch Ride-Through

The VNX2 platform introduces a 15-second de-bouncing timeout, during which the array uses its built-in batteries. A power failure is not reported and Multicore Cache does not start a dump to the vault.

This feature is especially useful during the transition from grid-provided power to generator-provided power, because it sometimes introduces a power flicker.

Multicore Cache Summary Example

To summarize the Multicore Cache features, a simple experiment was run with the following two arrays:

- VNX7600 array, running VNX2 OE 33 (beta build 687)
- VNX5500 array running VNX OE 32 (patch 201)

The same exact workload was run on the same sized drives. Refer to the description of the workload and setup notes in Appendix B: Unisphere Analyzer.

Cache Dirty Pages

Figure 14 displays a comparison3 between FLARE and MCx Cache Dirty Pages metrics. Note the following differences:

- FLARE cache maxed out quickly, and started force flushing
- Multicore Cache filled up very fast as well, but did not allow this workload to occupy 100% of cache
    - There was no forced flushing
    - Write throttling controlled dirty pages, until the array learned good pre-cleaning age value

[3] VNX2 Unisphere Analyzer does not show Cache Dirty Pages as a percent of all Cache. Multicore Cache shows actual MB used, but for the purpose of this paper, the data is represented as percentage. Refer to the real Analyzer graph output in Appendix B: Unisphere Analyzer.

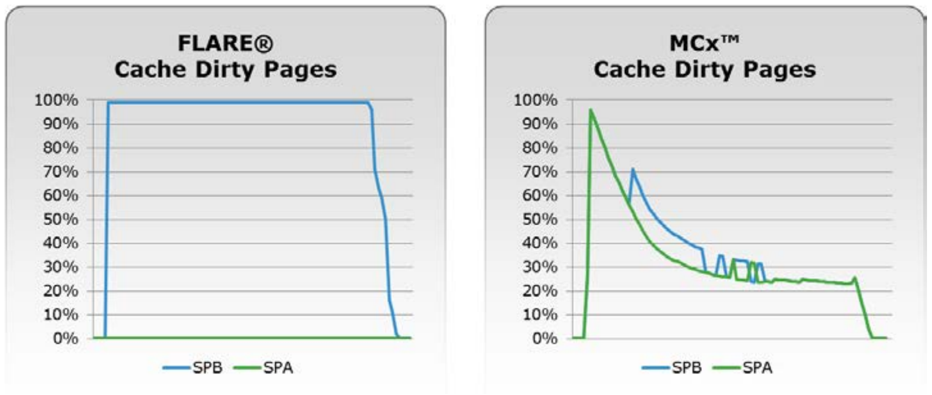


Figure 14. Cache dirty pages

Before dirty pages occupied the entire space, Multicore Cache started to throttle the workload, lowering the amount of the dirty cache pages. The VNX2 array was using both Storage Processors (Active/Active), and not just a single SP as with the FLARE system (Asymmetric Active/Active).

Storage Processors in the VNX2 array report dirty pages separately. Further, the dirty page number itself is estimated. At any time, the dirty page count should be the same between SPs, any variance being caused by the fact that both calculations are estimations. When Multicore Cache had learned enough about the workload, the dirty pages plateaued at about 25percent. At the same time, Multicore Cache stopped throttling. Other workloads, RAID group sizes, and LUN sizes would have plateaued at a different level, 25percent is not an expectation.

LUN Throughput

Examine the IOPs. As shown in Figure 15, MCx outperformed FLARE. Both graphs have the same Y-axis scale for easier comparison. However, the initial throughput of the MCx LUN was 27,000 IOPs, which occurred because all incoming writes hit DRAM Cache.

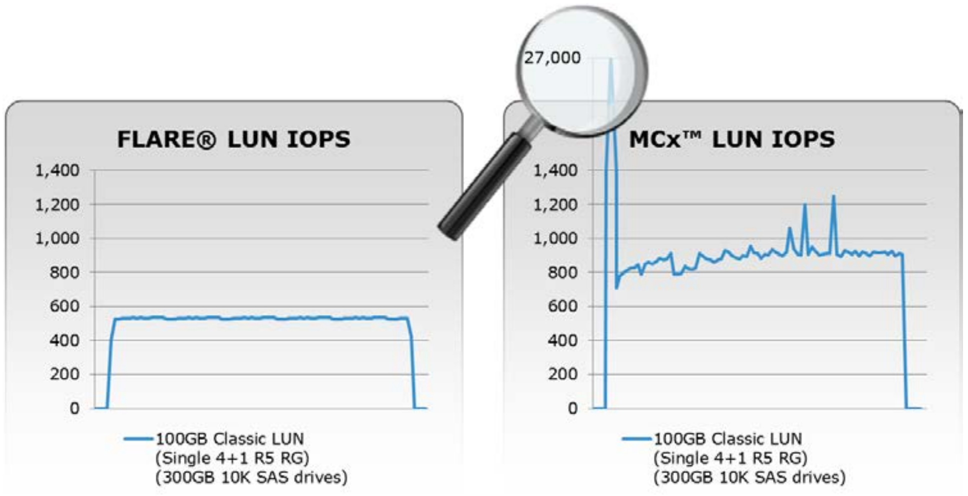


Figure 15. Throughput comparison

FLARE also had a brief moment when all incoming writes were hitting DRAM, except that the size of the available write cache was drastically different between the two systems. Unfortunately, the FLARE high throughput was too brief (for this test, the sample period was set at 60 seconds), and Analyzer did not reflect it. To make both tests comparable, the 100GB LUN was chosen precisely to exclude the physical DRAM size difference between the arrays.

The average throughput is noticeably different, even after dismissing the initial peak. MCx averaged at about 900IOPS, while FLARE averaged about 513IOPS—about a 150 percent improvement. Note that the drives used for this test were identical[4]: 300GB SAS 10K drives.

[4] The drives had different firmware levels. VNX OE release 33 uses newer firmware than the drives supported with VNX OE release 32 and below.

Symmetric Active/Active

A major component of MCx is the support for *Symmetric Active/Active* technology, which enables hosts to access LUNs by using both VNX Storage Processors simultaneously. With Symmetric Active/Active, VNX2 is transitioning away from the LUN ownership model and the inherently asymmetric performance it exalts.

All VNX arrays (including CLARiiON arrays running FLARE release 26 and later) support *Asymmetric Active/Active*, which is commonly and erroneously referred to as 'ALUA'.

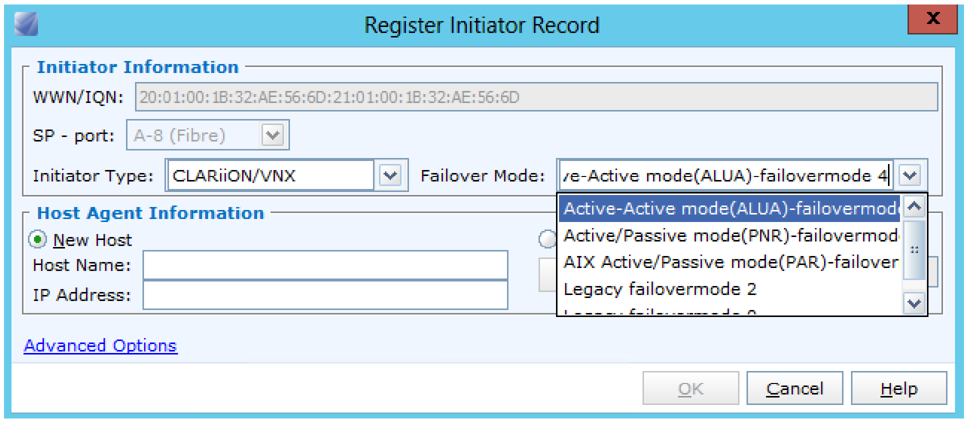
Asymmetric LUN Access or ALUA is a SCSI interface [2] that allows description of *Optimized* and *Non-Optimized* paths. The word 'asymmetric' in ALUA is misleading, and perhaps unfortunate. ALUA mode is simply the mechanism the SCSI protocol uses to communicate which paths should be used. Symmetric Active/Active uses ALUA messaging to signify that all paths are optimized. Asymmetric Active/Active uses ALUA to signify that some paths are optimized and some are non-optimized.

MCx supports Symmetric Active/Active, but it also supports Asymmetric Active/Active. Symmetric Active/Active is supported for Classic LUNs. A *Classic LUN* is a LUN that is not created within a Pool.

Note: This paper uses 'Active/Active' term to indicate Symmetric Active/Active, and Asymmetric Active/Active for the other ALUA-based access.

The VNX2 does not support Active/Active for Pool LUNs (both thick and thin). They default to Asymmetric Active/Active.

Because Active/Active is based on ALUA, it uses the same 'Failover mode 4', which is set at the initiator level during initiator registration, shown in Figure 16.



WWN/IQN: 20:01:00:1B:32:AE:56:6D:21:01:00:1B:32:AE:56:6D

SP - port: A-8 (Fibre)

Initiator Type: CLARiiON/VNX

Failover Mode: /e-Active mode(ALUA)-failovermode 4

Active-Active mode(ALUA)-failovermod
Active/Passive mode(PNR)-failovermod
AIX Active/Passive mode(PAR)-failover
Legacy failovermode 2

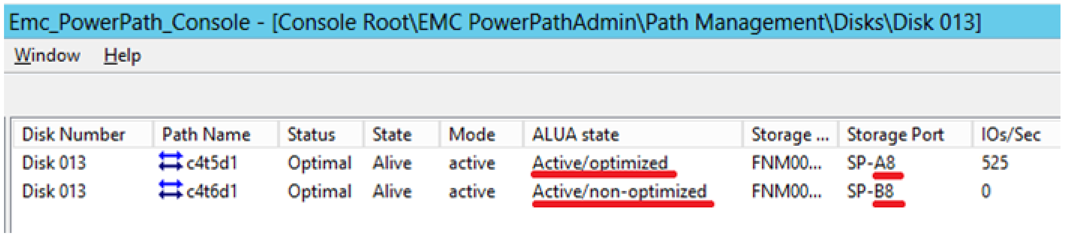
Host Name:

IP Address:

Figure 16. Active/Active failover mode

Active/Active and Multi-Pathing Software

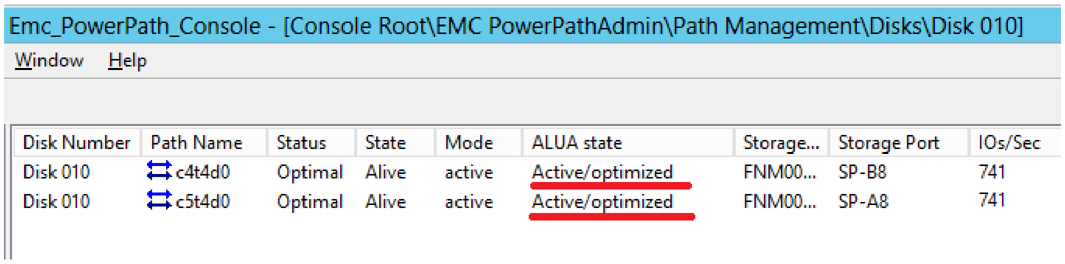
The client host and its SAN multipath software must recognize whether Symmetric Active/Active or Asymmetric Active/Active is used. EMC Power Path 5.7 and above supports Active/Active with the VNX2. Refer to Figure 17 and Figure 18 to see how Power Path 5.7 presents both cases.


Emc_PowerPath_Console - [Console Root\EMC PowerPathAdmin\Path Management\Disks\Disk 013]

Window Help

Disk Number	Path Name	Status	State	Mode	ALUA state	Storage ...	Storage Port	IOs/Sec
Disk 013	c4t5d1	Optimal	Alive	active	Active/optimized	FNM00...	SP-A8	525
Disk 013	c4t6d1	Optimal	Alive	active	Active/non-optimized	FNM00...	SP-B8	0

Figure 17. EMC Power Path and Asymmetric Active/Active


Emc_PowerPath_Console - [Console Root\EMC PowerPathAdmin\Path Management\Disks\Disk 010]

Window Help

Disk Number	Path Name	Status	State	Mode	ALUA state	Storage...	Storage Port	IOs/Sec
Disk 010	c4t4d0	Optimal	Alive	active	Active/optimized	FNM00...	SP-B8	741
Disk 010	c5t4d0	Optimal	Alive	active	Active/optimized	FNM00...	SP-A8	741

Figure 18. EMC Power Path and Symmetric Active/Active

Note the field order in the above figures is re-arranged from a default view. In default view, the ALUA state field appears as the right most field in the console.

Power Path recognizes Active/Active on a per-device basis. This way, the same host may have a Classic LUN and a Pool LUN provisioned.

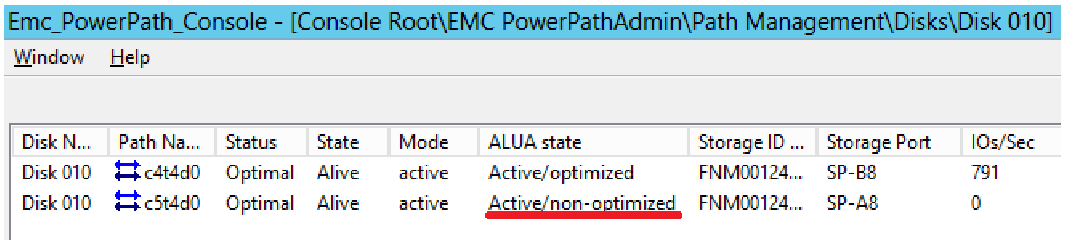
```
C:\> powermt display alua dev=all

Pseudo name=harddisk??
VNX ID=FNM00123456789 [Win2012]
Logical device ID=600601600980320002A012AC7E0F3E211 []
state=dead; policy=CLAROpt; queued-IOs=0
Owner: default=SP A, current=SP A       Array failover mode: 4
======================================================================
   - Host --    - Stor -    ------------ I/O Path -----------   -  Stats
###  I/O Paths   Interf.    ALUA State            Mode     State  Errors
======================================================================
   5 c5t5d1      SP A8      Active/optimized      active   active    0
   4 c4t5d1      SP B8      Active/non-optimized  active   active    0

Pseudo name=harddisk??
VNX ID=FNM00123456789 [Win2012]
Logical device ID=600601600980320002C012AC7E0F3E211 []
state=dead; policy=CLAROpt; queued-IOs=0
Owner: default=SP B, current=SP B       Array failover mode: 4
======================================================================
   - Host --    - Stor -    ------------ I/O Path -----------   -  Stats
###  I/O Paths   Interf.    ALUA State            Mode     State  Errors
======================================================================
   5 c4t4d0      SP A2      Active/optimized      active   active    0
   4 c4t4d0      SP B2      Active/optimized      active   active    0
```

CLI 2. Power Path display ALUA state

Classic LUNs support Active/Active, unless they use Data Services.

Note: The RecoverPoint splitter supports Active/Active.

When a snapshot of a classic LUN occurs, the VNX2 array immediately switches SCSI target port groups for this LUN to Asymmetric Active/Active. Cloning, MirrorView, and other data services, will have the same effect. With EMC Power Path 5.7 and above, the change does not require a SCSI bus rescan[5], as seen in Figure 19.


Emc_PowerPath_Console - [Console Root\EMC PowerPathAdmin\Path Management\Disks\Disk 010]

Disk N...	Path Na...	Status	State	Mode	ALUA state	Storage ID ...	Storage Port	IOs/Sec
Disk 010	c4t4d0	Optimal	Alive	active	Active/optimized	FNM00124...	SP-B8	791
Disk 010	c5t4d0	Optimal	Alive	active	Active/non-optimized	FNM00124...	SP-A8	0

Figure 19. Active/Active switches to Asymmetric Active/Active

Because of the Asymmetric Active/Active backward compatibility, every classic LUN keeps its Default SP Owner attribute. The attribute becomes relevant only if a classic LUN uses one of the advanced data services (such as SnapView), and the array must change to the Asymmetric Active/Active mode.

Locking Service

Active/Active is possible because MCx introduces a locking technology for LUN read and write access. SPs communicate with each other using the *Stripe Locking Service* with the *Communication Manager Interface* (CMI). The VNX2 hardware has a PCI Gen3 bus connecting both SPs, over which CMI runs between SPs.

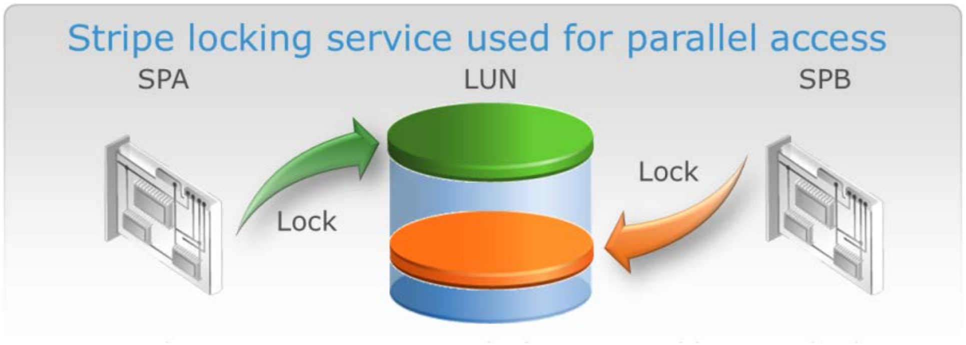


Figure 20. Stripe locking service

An SP must request an exclusive lock for a Logical Block Address (LBA) range on a LUN prior to proceeding with the write from a *Locking Service*. A locking service communication between SPs occurs over the CMI bus. The SP proceeds with the write upon a successful lock grant.

Similarly, a shared read lock must be taken out for reads. The SP needs to know whether to read from the disk or from a peer SP's cache. The peer SP may have newer data than the disk, because dirty cache pages have not been flushed from the peer SP to the disk yet. Once the I/O is complete, the lock is released.

[5] Please refer to EMC E-LAB Interoperability Navigator [3] for Active/Active compatibility with other multipath software solutions, such as Windows MPIO, Linux MPIO, and others.

Multicore FAST Cache

Multicore FAST Cache (or simply FAST Cache) identifies frequently accessed blocks and copies them from the spinning drives to flash drives. Data is copied into 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 Algorithm. VNX OE for Block version 05.33.009.5.155 supports SAS Flash or SAS Flash 2 Flash drives in FAST Cache.

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
- Support the Symmetric Active/Active access model

Functional Design

Multicore FAST Cache works with classic LUNs and pool LUNs. For more information about the VNX Pool structure, refer to the *EMC FAST VP white paper* [8].

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

Multicore FAST Cache Initial Warm Up

The first implementation of *FLARE FAST Cache* was introduced in Block Release 30. The warm up period for FAST Cache requires a 64KB block to be accessed three times before it is eligible to be copied to the flash media, as shown in Figure 21.

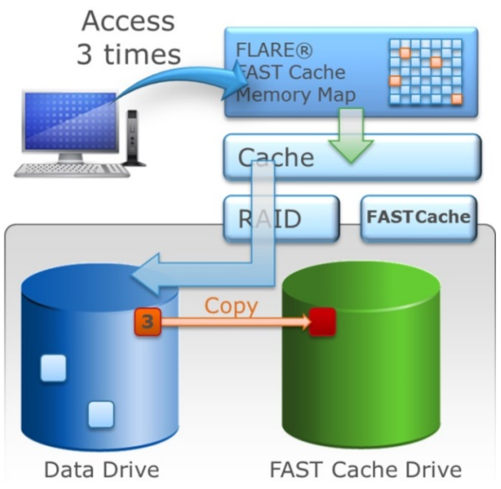


Figure 21. FLARE FAST Cache promotion

The warm up period depends entirely on the I/O intensity and the size of the work area. *Work area* is the range of LBA addresses on the LUN that is being addressed by a workload. For example, reading/writing to a single file within a small MS Access database means a small work area. Accessing an entire 2TB LUN corresponds to a large work area. The larger the work area, the smaller the chance of hitting the same 64KB block multiple times during a random workload.

Multicore FAST Cache utilizes the same 3-hit promotion pattern, illustrated in Figure 22. The difference between FLARE and MCx is the placement of Multicore FAST Cache within the system's stack order. Notice that Multicore FAST Cache is located below Multicore Cache, whereas FLARE FAST Cache was located above FLARE Cache. With MCx, the FLARE FAST Cache pass-through overhead to FLARE Cache is eliminated.

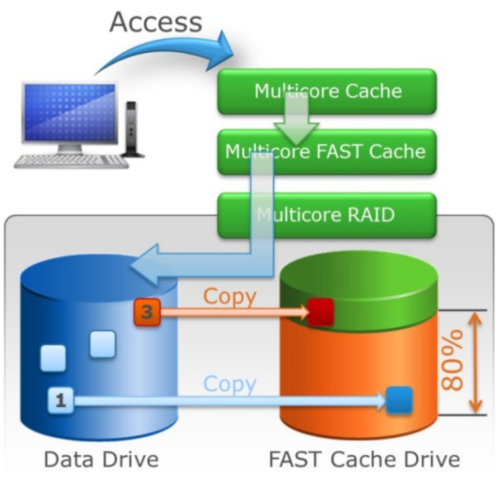


Figure 22. Multicore FAST Cache initial warm up

Page Management

Multicore FAST Cache operates with a page size of 64KB. When a page is first copied to FAST Cache, it is marked "clean." If it is modified by an incoming write IO, it becomes "dirty". The dirty pages are copied back to the data disk, and become clean again. The copy to the disk is an asynchronous process, and happens sometime after the host acknowledgement. When necessary, least recently used clean pages are overwritten by a new FAST Cache promotion from any FAST Cache enabled LUN.

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, as shown in Figure 23. 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.

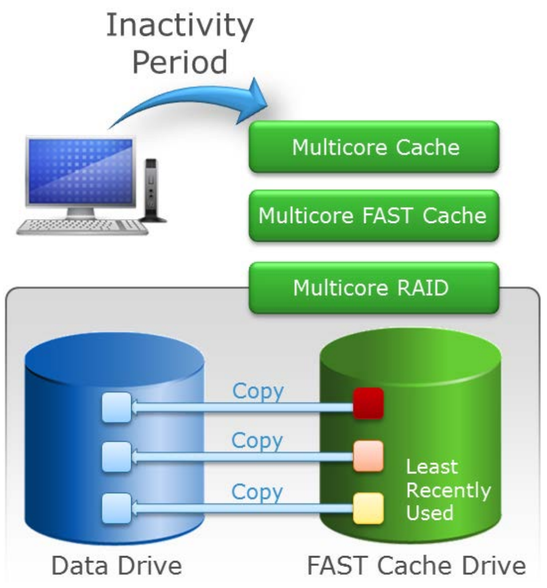


Figure 23. FAST Cache proactive cleaner

Keeping the amount of dirty pages low benefits FAST Cache. If a new workload is presented to the array, 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 require space, it first checks for available free or clean pages. If none is found, FAST Cache selects an appropriate amount of dirty pages using LRU algorithm, and copies them to data disks (flush operation) before reusing them for new promotions, as shown in Figure 24.

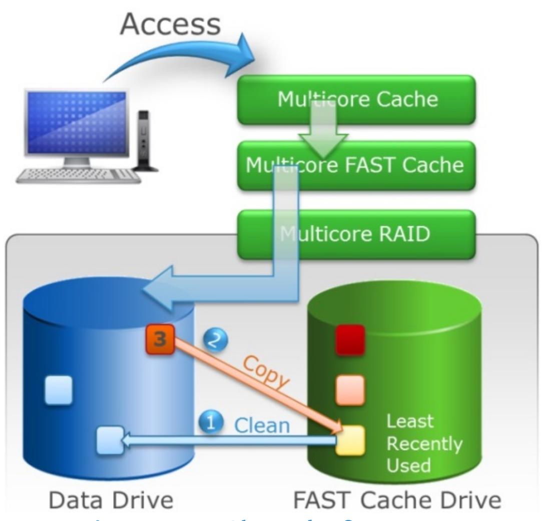


Figure 24. Clean before copy

Minimal Page Count

Multicore FAST Cache prevents any individual LUN from using all available pages at the expense of the other LUNs. A *Minimal Page Count* metric is used, which guarantees every LUN a fair share of FAST Cache space.

The minimum is calculated by equally sharing 50 percent of all Multicore FAST Cache pages used by all classic and private LUNs in the system. The other 50 percent is freely available for any LUN's use. If a LUN does not use all of its minimal count pages, they remain freely available to other LUNs. The minimal count is further adjusted to be no more than half the LUN's size. The count is a dynamic number and is adjusted every time a LUN is bound or destroyed.

When FAST Cache is 100 percent used, which is an intended state of a well-functioning cache, any new promotion requires a clean or a free page. FAST Cache chooses the pages so every LUN keeps its minimal page count for its most-used data.

Consider the example shown in Figure 25:

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.

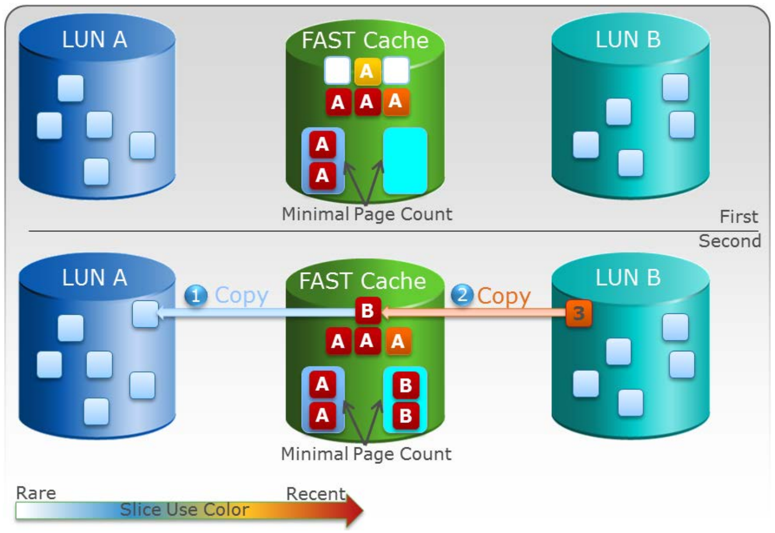


Figure 25. Multicore FAST Cache minimal page count

Then FAST Cache starts to promote LUN B's data. First two data blocks from LUN B obtain the two free pages. A third promotion from LUN B causes FAST Cache to flush a 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 LUNs were added to this system, the minimal page count would become one page per LUN.

FAST Cache Writes

A significant enhancement in this release is the slight rearrangement of the IO stack: the FAST Cache memory map is moved to be below Multicore Cache, rather than above Cache in FLARE, as shown in Figure 26. The benefit is that MCx is quicker to acknowledge a host write than the previous FAST Cache implementation.

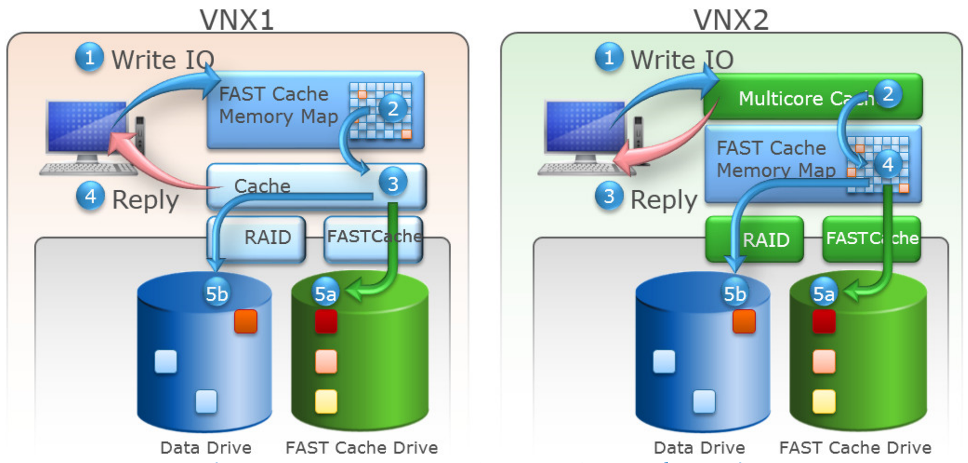


Figure 26. FLARE vs. MCx FAST Cache write

In FLARE, Write I/O (1) is first checked against FAST Cache memory map (2), and then it lands in Cache (3). A write acknowledgement follows (4). When Cache is ready to flush the page, it sends the data to FAST Cache (5a) if memory map has a FAST Cache pointer, or to data drive (5b) if there is no FAST Cache record.

In MCx, Write I/O (1) lands in 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 sends the page to FAST Cache, which checks the memory map (4), and depending whether the data belongs to FAST Cache or not sends it to the appropriate location: FAST Cache (5a) or data drive (5b).

MCx acknowledges the write I/O faster, has a larger Cache, and the data spends more time in RAM when compared to the FLARE-based implementation.

If Multicore Cache's write function is disabled, the incoming write acknowledgement is delayed until the I/O is saved either to Multicore FAST Cache flash drive, or to data drive.

FAST Cache Reads

Read operations have also changed with MCx, but the overall effect is less noticeable by the hosts—the number of steps for a read I/O remain the same between VNX1 and the VNX2, as shown in Figure 27.

In VNX running FLARE, Read I/O (1) is first checked against FAST Cache memory map (2). If there is a FAST Cache pointer, the data is read from FAST Cache (3a), which reads the page(s) from FAST Cache drives (4a). If the data does not have a FAST Cache memory map record, the read request passes to Cache (3b), which fulfills it from the data already in cache if it is a *read hit* or from data drive (4b) if it is a *read miss*. The data is loaded to Cache, and the host receives a reply (5).

In the VNX2 running with MCx, Read I/O (1) first lands in Multicore Cache (2). A read hit condition will cause an immediate reply to the host. A read miss condition will result in FAST Cache memory map being checked (3). If a FAST Cache record is found, the data is read from FAST Cache drive (4a) or from 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).

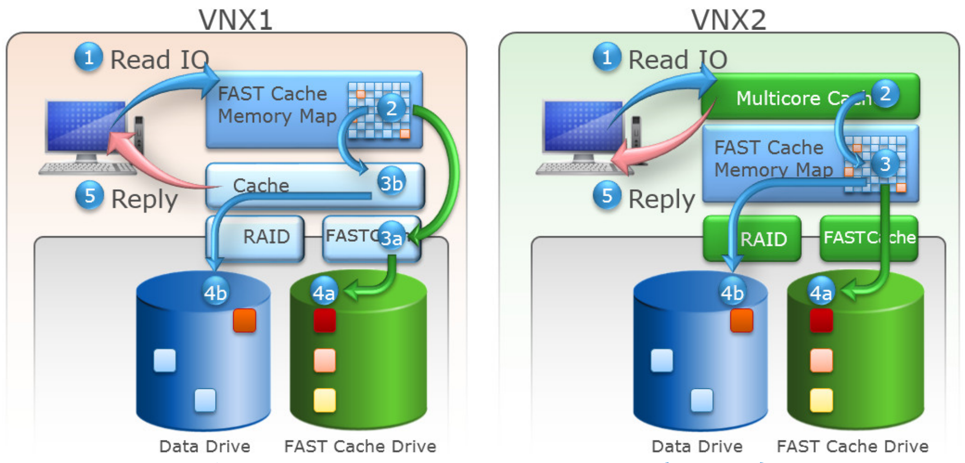


Figure 27. FLARE vs. MCx FAST Cache read

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.

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

The two features (Cache and FAST Cache) have their own strengths, and they allow each other to service corresponding I/O, improving the overall system performance, and simplifying management.

Multicore FAST Cache Configuration Options

The VNX2 series arrays have a maximum Multicore FAST Cache value of 4.8TB (VNX8000). FAST Cache only supports RAID1 (mirrored) pairs; Multicore FAST Cache must be configured with an even number of drives. FAST Cache forms RAID1 pairs based on the number of drives provided. VNX OE for Block version 05.33.009.5.155 supports SAS Flash or SAS Flash 2 Flash drives in FAST Cache.

The following tables below describe the maximum FAST Cache capacities when using SAS Flash or SAS Flash 2 Flash drives:

SAS Flash Maximum FAST Cache Capacities

Model	**Max FAST Cache capacity (rounded up)**	**Number* of 100/200 GB Flash drives****
VNX8000	100GB drives: 2,100GB 200GB drives: 4,800GB	From 2 to 48
VNX7600	100GB drives: 2,100GB 200GB drives: 4,200GB	From 2 to 42
VNX5800	100GB drives: 1,500GB 200GB drives: 3,000GB	From 2 to 30
VNX5600	100GB drives: 1,000GB 200GB drives: 2,000GB	From 2 to 20
VNX5400	100GB drives: 500GB 200GB drives: 1,000GB	From 2 to 10
VNX5200	100GB drives: 300GB 200GB drives: 600GB	From 2 to 6

** Use even number of drives
*   Do not mix drive sizes

SAS Flash 2 Maximum FAST Cache Capacities

Model	Max FAST Cache capacity (rounded up)	Number* of 200/400GB Flash drives**
VNX8000	200GB drives: 2,000GB 400GB drives: 4,800GB	From 2 to 24
VNX7600	200GB drives: 2,000GB 400GB drives: 4,000GB	From 2 to 20
VNX5800	200GB drives: 1,400GB 400GB drives: 2,800GB	From 2 to 14
VNX5600	200GB drives: 1,000GB 400GB drives: 2,000GB	From 2 to 10
VNX5400	200GB drives: 1,000GB 400GB drives: 800GB	From 2 to 10
VNX5200	200GB drives: 600GB	From 2 to 6

** Use even number of drives
* Do not mix drive sizes

Note: EMC suggests utilizing SAS Flash drives in Multicore FAST Cache for configurations of 600 GBs and less.

Multicore RAID

Multicore RAID is the MCx component that defines, manages, creates, and maintains VNX2 RAID protection. It replaces the FLARE RAID engine.

Multicore RAID scales across multiple CPU cores. The more cores the physical Storage Processor has, the more processing power is available for the RAID engine. MCx RAID operations are not isolated to CPU0.

Multicore RAID includes significant improvements—such as Permanent Sparing, Drive Mobility, Hot Spare policies, and Improved RAID durability—that 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 drives. When a drive fails, Multicore RAID *spares* it to a suitable unbound drive, and that new drive becomes a permanent part of the RAID group. When a failed drive is replaced, the new drive simply becomes unbound and available to be a future spare.

In the same scenario, FLARE would have *equalized* the hot spare with a replacement drive, which means that it would copy data back from a hot spare to the new drive. Equalization is a FLARE requirement because it keeps RAID group structure bound to the drive slot numbers. When a hot spare is invoked, a new temporary slot is introduced into the RAID structure.

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

As shown in Figure 28, a drive from a given RAID group (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 RAID group and the Multicore RAID engine rebuilds the drive. After the rebuild, the RAID group will operate normally.

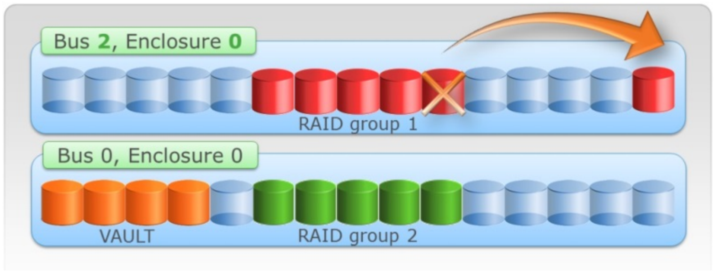


Figure 28. Permanent sparing

In certain environments, customers may still need physical positions fixed for RAID groups. For example, some drives and/or Disk Array Enclosures (DAEs) may be bound for specific cost centers and applications. Hence, a failed drive will need to be replaced in the same location to maintain the association. Another reason for the rigid positioning might be government regulations, where rules require very specific hardware mapping. If there is a potential security breach, all drives containing sensitive data can quickly and easily be removed by anyone according to a pre-designed RAID group mapping affixed to the front panel of the array or the cabinet.

For these scenarios, the CLI[6] ability to initiate a manual copy back (also referred to as a *Copy-To-Disk* operation) is still available. The copy back is very similar to an equalize operation, but must be started manually. The command below initiates a copy operation from the disk located in slot 2_0_14 to the disk located in slot 2_0_8.

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0
copytodisk 2_0_14 2_0_8


WARNING: The data from disk 2_0_14 will be copied to disk 2_0_8. This
process cannot be aborted and may take a long time to complete.

Do you wish to continue (y/n)? y
```

CLI 3. Copy back to a specific disk

Drive Mobility

Drive Mobility, also known as *Portable Drives*, allows users to move VNX2 disks within the array. 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 VNX arrays. Every array keeps track of its own disks. All new and unknown drives are automatically formatted and zeroed.

Non-vault drives can be moved to any slot within the array, as long as they fit within available space. Figure 29 shows a red RAID group that has two drives moved to new locations, including a different enclosure.

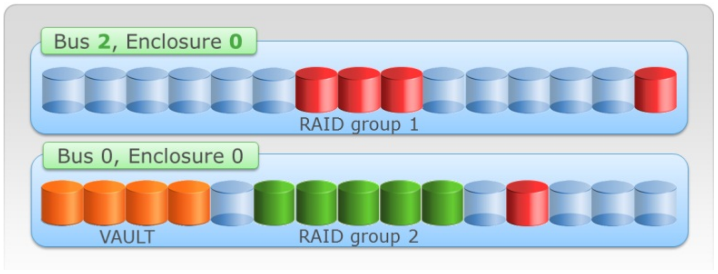


Figure 29. Drive mobility

FLARE identifies drives by their physical address, for example, Bus 1 Enclosure 0 Disk 11 or in shortened form, 1_0_11. When a drive is moved from 1_0_11 to 2_1_13, FLARE does not recognize the move, it simply notes that a drive from 1_0_11 has been removed, and a new drive in 2_1_13 was found. Consequently, any data that

[6] Unisphere provides an ability to initiate a similar disk copy operation, but without the option of choosing a destination drive. When started, Unisphere picks a suitable unbound drive for the data copy. The algorithm to choose the drive is exactly the same as when choosing a replacement for a failed drive.

was on the disk in 1_0_11 is not recognized by FLARE in 2_1_13. Multicore RAID memorizes drives by their serial number, and easily understands drive movement.

Multicore RAID allows online drive movement. RAID group data continues to be available after a drive is pulled out. When a drive becomes unavailable (for example, it fails, is pulled out of the slot, encounters a firmware error, and so on.) MCx starts a 5-minute counter. Drive sparing is invoked after the 5-minute interval passes. See the Hot Spares section for more details.

Pulling any active drive out degrades its RAID group. If a RAID group 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 array, the RAID group experiences a "Data Unavailable" event and become faulted.

Note: Move drives within VNX2 arrays with caution.

After the drive is re-inserted into the array, the RAID group returns to a healthy, operational state and no data is lost.

DAE Re-Cabling

Multicore RAID drive mobility also optimizes availability of back-end buses. DAE re-cabling does require a power down, but because of Drive Mobility, re-cabling can now occur with all data preserved. Users can simply power the array down, cable DAEs as needed (typically utilizing more back-end buses; the EMC specialist leading the procedure will provide guidance and expertise), and power the array back up. Upon power up, the array will easily locate the drives by surveying their serial numbers, validating all RAID group members, and continuing operations.

Some customers like to re-arrange their entire pools or several RAID groups for various reasons, for example, adding more DAEs and back-end buses. Although, it is possible to rearrange all drives while the array is up and running, it is safer, and faster to do mass-rearrangements when the array is shut down. Upon restart, the new drive locations are noted, and no other reconfigurations are needed.

Hot Spares

Multicore RAID changes the traditional VNX Hot Spare paradigm, because there is no need to define hot spares on the array. In fact, with MCx it is no longer possible to define hot spares. MCx spares failed drives to compatible[7] unbound drives.

Upon any drive failure (excluding the system drives), Multicore RAID will:

- Wait 5 minutes
    - The timeout gives a failed drive a chance to recover
- After the 5 minutes, look for a suitable replacement in the array among unbound drives
- When and if a match is found:

[7] In FLARE, situations occur when a larger failed drive is spared to a smaller drive. This is possible only when the larger drive is not fully used, for example, its used space is small enough to fit on a smaller drive. Multicore RAID does not support this option. The spare drive's size is determined by the RAID group boundaries

- o Add the new drive to the RAID group
- o Start the drive rebuild process

The 5-minute interval avoids unnecessary full drive rebuilds, which could last up to several days. The VNX2 array easily survives human errors, when a wrong drive is pulled during a routine failed drive replacement procedure. Insert the drive back within a 5-minute window, and avoid a timely rebuild. Soft recoverable error conditions also may now be fixed without the need to invoke a full drive rebuild.

Drive Rebuild

Multicore RAID rebuilds only used sections of drives, one LUN at a time, top to bottom. Any unused space is not rebuilt, as there is nothing to rebuild.

For example in Figure 30, the RAID group has two LUNs: 0 and 1. Therefore, each drive in a RAID group houses a portion of each LUN. Upon rebuild, the LUN 0 area of the new drive will be rebuilt first, followed by LUN 1 area.

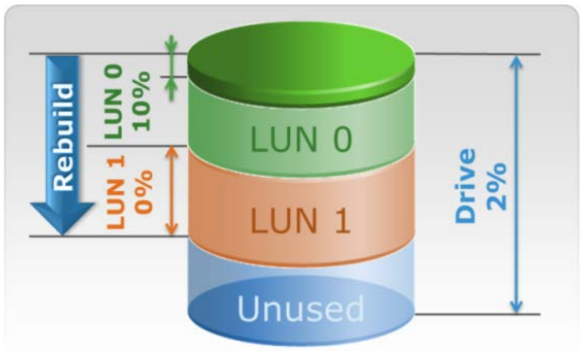


Figure 30. Drive rebuild

To check drive 1_1_11 rebuild progress (percentage based) from the CLI, use the following command:

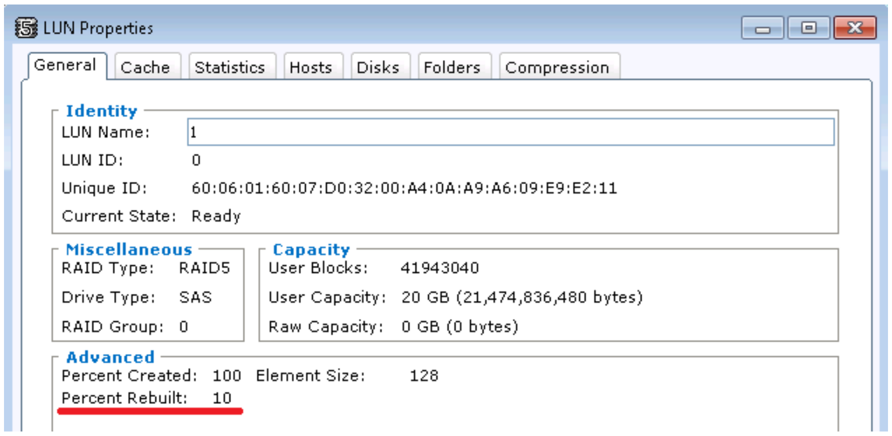
```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0
getdisk 1_1_11 -rb

Bus 1 Enclosure 1  Disk 11
Prct Rebuilt:            0: 10 1: 0
```

CLI 4. Drive percent rebuilt

The example above, shows that the LUN 0 area has just started to be rebuilt (10 percent complete), and LUN 1 is waiting for its turn (0 percent complete). The LUN 1 rebuild process starts after LUN 0 is 100 percent complete.

If the second drive (for example, in a RAID6 group) fails, the LUN percent rebuilt (`Prct rebuilt`) statistic will be adjusted to include the LUN's space on the second drive. The combined space needing to rebuild a LUN would double (two drive's worth), therefore the percent rebuilt is reduced by half.

Unisphere shows individual LUN rebuild progress as part of the LUN Properties, as seen in Figure 31.



Identity

LUN Name:	1
LUN ID:	0
Unique ID:	60:06:01:60:07:D0:32:00:A4:0A:A9:A6:09:E9:E2:11
Current State:	Ready

Miscellaneous

RAID Type:	RAID5
Drive Type:	SAS
RAID Group:	0

Capacity

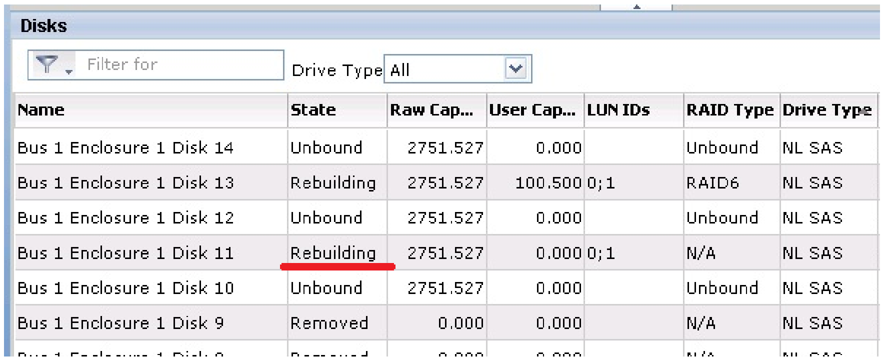
User Blocks:	41943040
User Capacity:	20 GB (21,474,836,480 bytes)
Raw Capacity:	0 GB (0 bytes)

Advanced

Percent Created:	100	Element Size:	128
Percent Rebuilt:	10		

Figure 31. LUN percent rebuilt

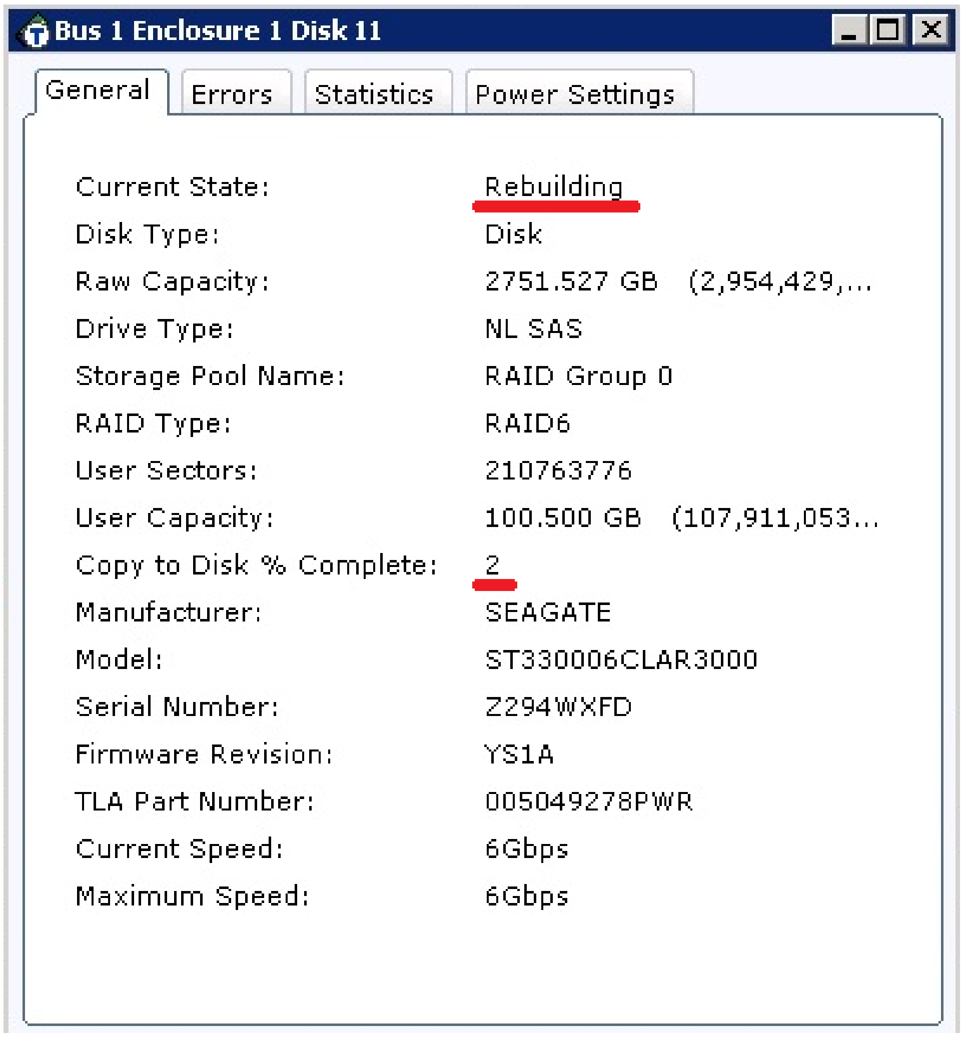
Unisphere also shows the drive state as Rebuilding. Navigate to System › Hardware › Disks and in the lower half of the window, find a list of all disks present in the array, as shown in Figure 32 below.



Name	State	Raw Cap...	User Cap...	LUN IDs	RAID Type	Drive Type
Bus 1 Enclosure 1 Disk 14	Unbound	2751.527	0.000		Unbound	NL SAS
Bus 1 Enclosure 1 Disk 13	Rebuilding	2751.527	100.500	0;1	RAID6	NL SAS
Bus 1 Enclosure 1 Disk 12	Unbound	2751.527	0.000		Unbound	NL SAS
Bus 1 Enclosure 1 Disk 11	Rebuilding	2751.527	0.000	0;1	N/A	NL SAS
Bus 1 Enclosure 1 Disk 10	Unbound	2751.527	0.000		Unbound	NL SAS
Bus 1 Enclosure 1 Disk 9	Removed	0.000	0.000		N/A	NL SAS

Figure 32. Rebuilding disk

To get more information about the process, view the disk properties (by right clicking the disk, and selecting Properties). View the "Copy to Disk % Complete" metric. Look at Figure 33 below:



Current State:	Rebuilding
Disk Type:	Disk
Raw Capacity:	2751.527 GB (2,954,429,...
Drive Type:	NL SAS
Storage Pool Name:	RAID Group 0
RAID Type:	RAID6
User Sectors:	210763776
User Capacity:	100.500 GB (107,911,053...
Copy to Disk % Complete:	2
Manufacturer:	SEAGATE
Model:	ST330006CLAR3000
Serial Number:	Z294WXFD
Firmware Revision:	YS1A
TLA Part Number:	005049278PWR
Current Speed:	6Gbps
Maximum Speed:	6Gbps

Figure 33. Disk properties

The "Copy to disk % Complete" calculation is for the entire disk capacity. The illustration shows that 2percent of the drive capacity has been rebuilt, which is about 55GB of space for a 3TB drive (2,751GB usable space).

The CLI shows the same information:

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0
getdisk 1_1_11 –all

< --- several lines abridged --- >
Prct Copied:              2
```

CLI 5. Drive percent copied

The LUN progress is calculated differently. The LUN "Percent rebuilt" number shows the statistic from the LUN's point of view. Previous CLI `getdisk` output showed that LUN 0 was 10 percent rebuilt, therefore 55GB matched 10 percent of the LUN 0 area to be rebuilt on the new drive.

Rebuild Mathematics Explained

In a test system, drive 1_1_11 was part of a RAID6 4+2 RAID group. As a result, every LUN on this RAID group was split across four disks (two[8] disks had parity).

The following steps show a full calculation (Refer to Figure 30):

- 2% total disk rebuilt equals 2751GB * 2% = 55GB
- LUN 0 occupies about 550GB (10% LUN0 * 55GB) on 1_1_11 Disk
- LUN 0's full size is 4*550GB=2200GB (the number of data drives in the RAID group multiplied by the LUN 0 area on a single disk)

Sparing Rules

Multicore RAID introduces a small, but helpful change to the *VNX Sparing Rules*. The sparing rules define the search order and logic for locating the best suitable drive to replace a failed disk. Multicore RAID adds an enclosure rule that prefers using a drive within the same enclosure before using a drive in another enclosure.

The array uses the following logic depicted in Figure 34 to select a suitable drive replacement:

1. Type—the array looks for all available drives of the same Type (SAS Flash, SAS Flash 2, SAS Flash 3, SAS, NL-SAS, etc.) as the failed drive.
2. Bus—from the drives found, the array then selects the drives on the same Bus.
3. Size— from the drives found, the array then selects the drives of the same RAID group capacity (or larger).
4. Enclosure—and finally, the system checks whether any of the drives found are in the same Enclosure as the failed drive.
   - The array selects one drive, if the final set has several drives.

Note: The above logic is weighted from highest to lowest weight, with the Drive Type being the highest weight. The logic is specifically designed to select a replacement drive closest to the failed drive. For instance, a replacement drive of the same Type and Bus of a failed drive will always be chosen before a replacement on a different bus.

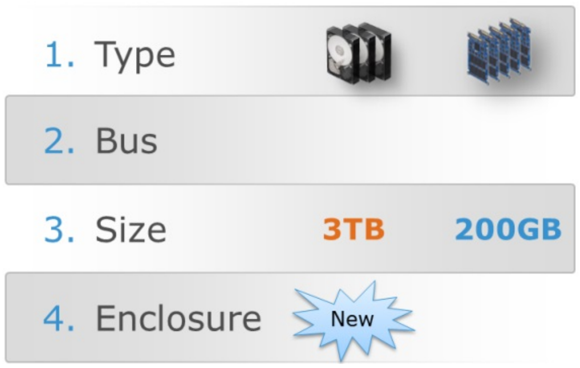


Figure 34. Sparing rules

[8] The true distribution is little more complicated for RAID6, as parity is also distributed, but the simplified approach works for the purposes of this example

As indicated earlier, sparing starts following a 5-minute delay after a drive goes offline. The *Proactive Sparing* technology, which evacuates data off a drive that is about to fail, does not incur the 5-minute delay. The array administrator's *Manual User Copy* ability to initiate a manual copy to a new drive also does not have the 5-minute delay.

Suitable Replacement Drive

When selecting a suitable drive Multicore RAID does not always use the size of the failed drive. It will use a drive that is better aligned with the original RAID group size.

For example, a drive used in the RAID group built from 300GB SAS drives fails. The only suitable replacement present is a 900GB SAS drive, which rebuilds the data and wastes about 600GB of usable capacity. When a new 300GB drive replaces the failed drive, it becomes unbound.

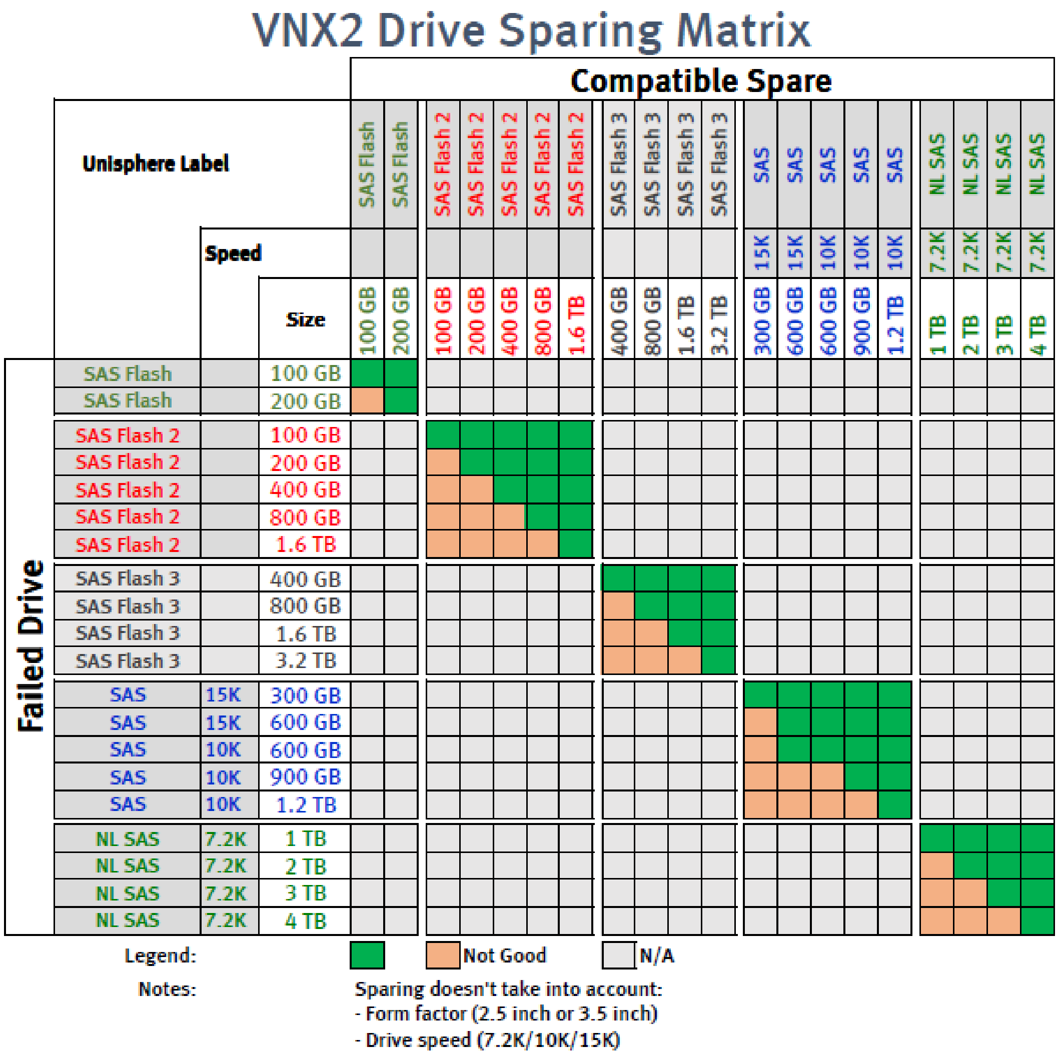
A good way to bring the new 300GB drive into the RAID group and to free the underused 900GB drive is to start a manual Copy-To-Disk operation, using the CLI command. Alternatively, if (at some time in the future) the 900GB drive also fails, Multicore RAID will look for a suitable 300GB drive, not another 900GB drive.

Drive Sparing Matrix

The VNX2 series arrays support a broad range of drive types to meet customers' business demands. The wide selection requires the implementation of numerous compatibility rules. Appendix D: VNX2 Flash Drive Technology outlines the various drive technologies used for each of the Flash drives supported in the VNX2 system.

For example, the latest Flash drives supported on VNX2 have three families: SAS Flash, SAS Flash 2, and SAS Flash 3 respectively. Although these may seem nearly identical, they differ significantly on an underlying technology level. These 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. In VNX OE for Block version 05.33.009.5.155 SATA Flash has been renamed to SAS Flash, and SAS Flash VP is renamed to SAS Flash 2.

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



VNX2 Drive Sparing Matrix

Failed Drive: Unisphere Label	Speed	Size	SAS Flash	SAS Flash	SAS Flash 2	SAS Flash 2	SAS Flash 2	SAS Flash 2	SAS Flash 2	SAS Flash 3	SAS Flash 3	SAS Flash 3	SAS Flash 3	SAS	SAS	SAS	SAS	SAS	NL SAS	NL SAS	NL SAS	NL SAS
	Speed													15K	15K	10K	10K	10K	7.2K	7.2K	7.2K	7.2K
		Size	100 GB	200 GB	100 GB	200 GB	400 GB	800 GB	1.6 TB	400 GB	800 GB	1.6 TB	3.2 TB	300 GB	600 GB	600 GB	900 GB	1.2 TB	1 TB	2 TB	3 TB	4 TB
SAS Flash		100 GB	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash		200 GB	🟧	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 2		100 GB	⬜	⬜	🟩	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 2		200 GB	⬜	⬜	🟧	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 2		400 GB	⬜	⬜	🟧	🟧	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 2		800 GB	⬜	⬜	🟧	🟧	🟧	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 2		1.6 TB	⬜	⬜	🟧	🟧	🟧	🟧	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 3		400 GB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 3		800 GB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 3		1.6 TB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟧	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS Flash 3		3.2 TB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟧	🟧	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
SAS	15K	300 GB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜
SAS	15K	600 GB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜
SAS	10K	600 GB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜
SAS	10K	900 GB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟧	🟧	🟩	🟩	⬜	⬜	⬜	⬜
SAS	10K	1.2 TB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟧	🟧	🟧	🟩	⬜	⬜	⬜	⬜
NL SAS	7.2K	1 TB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩
NL SAS	7.2K	2 TB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟩	🟩	🟩
NL SAS	7.2K	3 TB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟧	🟩	🟩
NL SAS	7.2K	4 TB	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟧	🟧	🟧	🟩

Compatible Spare

Failed Drive

Legend: 🟩 🟧 Not Good ⬜ N/A

Notes: Sparing doesn't take into account:
- Form factor (2.5 inch or 3.5 inch)
- Drive speed (7.2K/10K/15K)

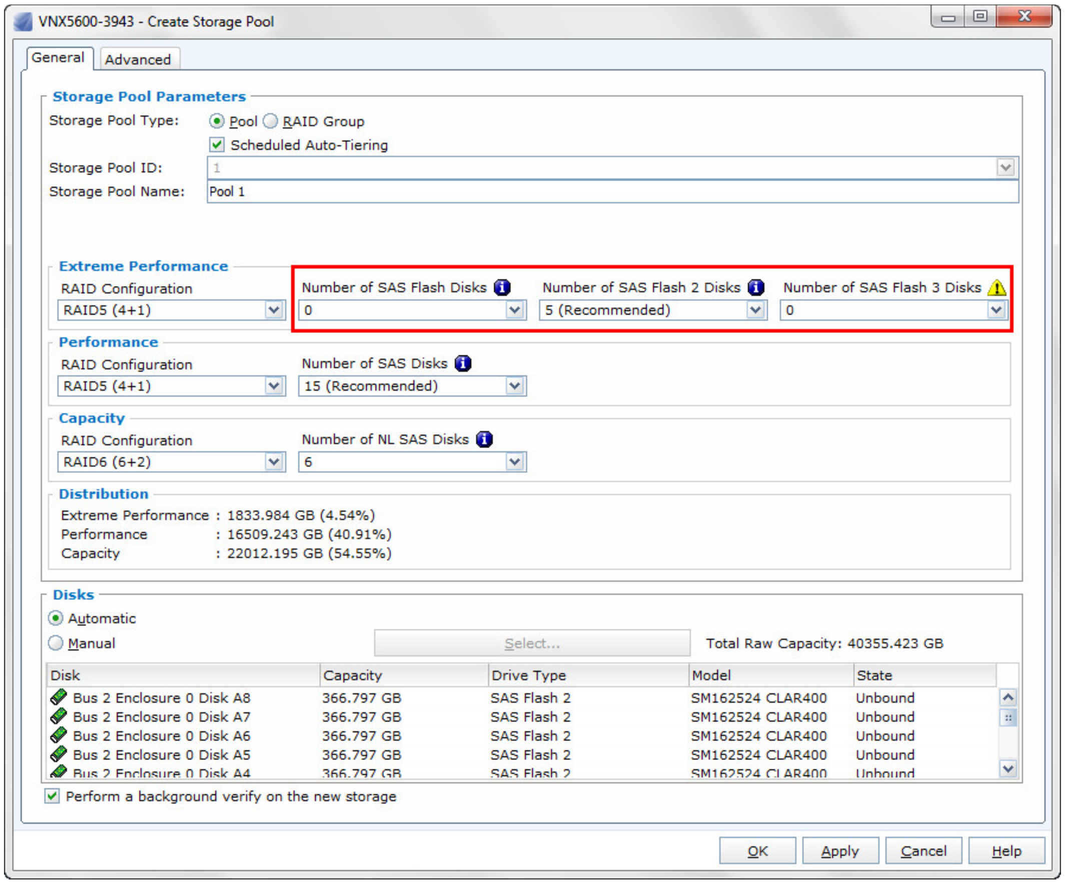
Figure 35. Drive sparing matrix

RAID Groups and Pools Drive Compatibility

Figure 35 shows groups of drives organized in five sets. The compatibility within each set is used for sparing and for RAID group creation. VNX does not allow RAID groups to contain drives from different sets. For example, SAS and NL SAS drives cannot be used within the same RAID group. SAS Flash, SAS Flash 2, and SAS Flash 3 are not allowed to mix either.

Pool tiers have a somewhat different approach. VNX pools support up to three tiers [7] and every tier claims a specific RAID group set, except Extreme Performance that claims two sets. Even though SAS Flash 2 drives cannot be configured in the same RAID group with SAS Flash drives, private RAID groups created from each set can be a part of the Extreme Performance tier within the same pool. In VNX OE for Block version 05.33.009.5.155 SAS Flash 3 drives are only supported in all flash Storage Pools with drives of the same type.

In Unisphere, when drives from SAS Flash, SAS Flash 2, and SAS Flash 3 are present, the pool creation Extreme Performance section includes three drop-down boxes, one for each set. Figure 36 below shows an example of the Storage Pool creation window when all 3 Flash drive type are present within the system.



VNX5600-3943 - Create Storage Pool

General | Advanced

Storage Pool Parameters

Storage Pool Type:	[✓ Pool] RAID Group
	[✓] Scheduled Auto-Tiering
Storage Pool ID:	1
Storage Pool Name:	Pool 1

Extreme Performance

RAID Configuration	Number of SAS Flash Disks	Number of SAS Flash 2 Disks	Number of SAS Flash 3 Disks
RAID5 (4+1)	0	5 (Recommended)	0

Performance

RAID Configuration	Number of SAS Disks
RAID5 (4+1)	15 (Recommended)

Capacity

RAID Configuration	Number of NL SAS Disks
RAID6 (6+2)	6

Distribution

```
Extreme Performance : 1833.984 GB (4.54%)
Performance         : 16509.243 GB (40.91%)
Capacity            : 22012.195 GB (54.55%)
```

Disks

[✓ Automatic]
Manual

Select...

Total Raw Capacity: 40355.423 GB

Disk	Capacity	Drive Type	Model	State
Bus 2 Enclosure 0 Disk A8	366.797 GB	SAS Flash 2	SM162524 CLAR400	Unbound
Bus 2 Enclosure 0 Disk A7	366.797 GB	SAS Flash 2	SM162524 CLAR400	Unbound
Bus 2 Enclosure 0 Disk A6	366.797 GB	SAS Flash 2	SM162524 CLAR400	Unbound
Bus 2 Enclosure 0 Disk A5	366.797 GB	SAS Flash 2	SM162524 CLAR400	Unbound
Bus 2 Enclosure 0 Disk A4	366.797 GB	SAS Flash 2	SM162524 CLAR400	Unbound

[✓] Perform a background verify on the new storage

Figure 36. Extreme Performance tier

Hot Spare Policy

Multicore RAID introduces a new concept to the VNX2 array. It is the ability to forewarn an administrator about the state of usable spare drives in the system.

For that purpose, Multicore RAID introduces *Hot Spare Policies.* A hot spare policy monitors a number of unbound drives (potential spare drives) of a certain type in the system, and posts a warning when that number is low.

There are three Hot Spare Policies:

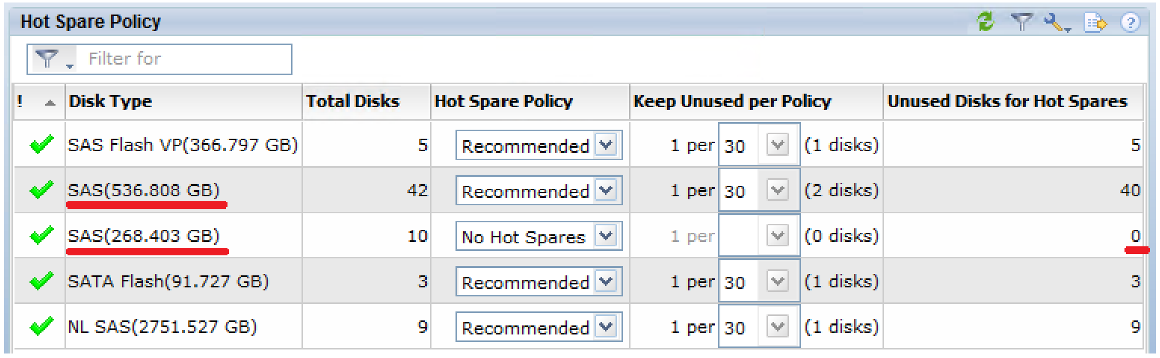
- *Recommended*—1 hot spare per 30 drives of every type
- *Custom*—Allow an administrator to specify the ratio
- *No Hot Spares*—Turn off policy monitoring

Note: The *No Hot Spares* policy does not stop sparing. A failed drive will always be spared, if a suitable unbound drive is found.

Note: In instances where the Recommended Policy does not match the Best Practice of 1 spare for every 30 drives, correct this by using the Custom Policy.

Multicore RAID defines a policy for every supported drive type and size, but does not distinguish drive speeds or form factors. For instance, Multicore RAID does not differentiate a 2.5-inch form factor 600GB SAS drive with a speed of 10K from a 3.5-inch 600GB SAS drive with 15K speed, where sparing is concerned. Both are considered to be the same, and therefore compatible for sparing.

Hot spare policies are specifically defined by drive types and capacity. For example, SAS 300GB drives and SAS 600GB drives each have independent hot spare policies defined, as shown in Figure 37. The VNX2 array, depicted in the figure below, has all ten SAS 300GB drives already used. The system does not show a warning because *No Hot Spares* policy is set. However, if one of the SAS 300GB drives fails, Multicore RAID will spare to the best suitable drive. In this case, it will be one of the 40 unbound SAS 600GB drives (Refer to Figure 35 for a full list of compatible drives).


Hot Spare Policy

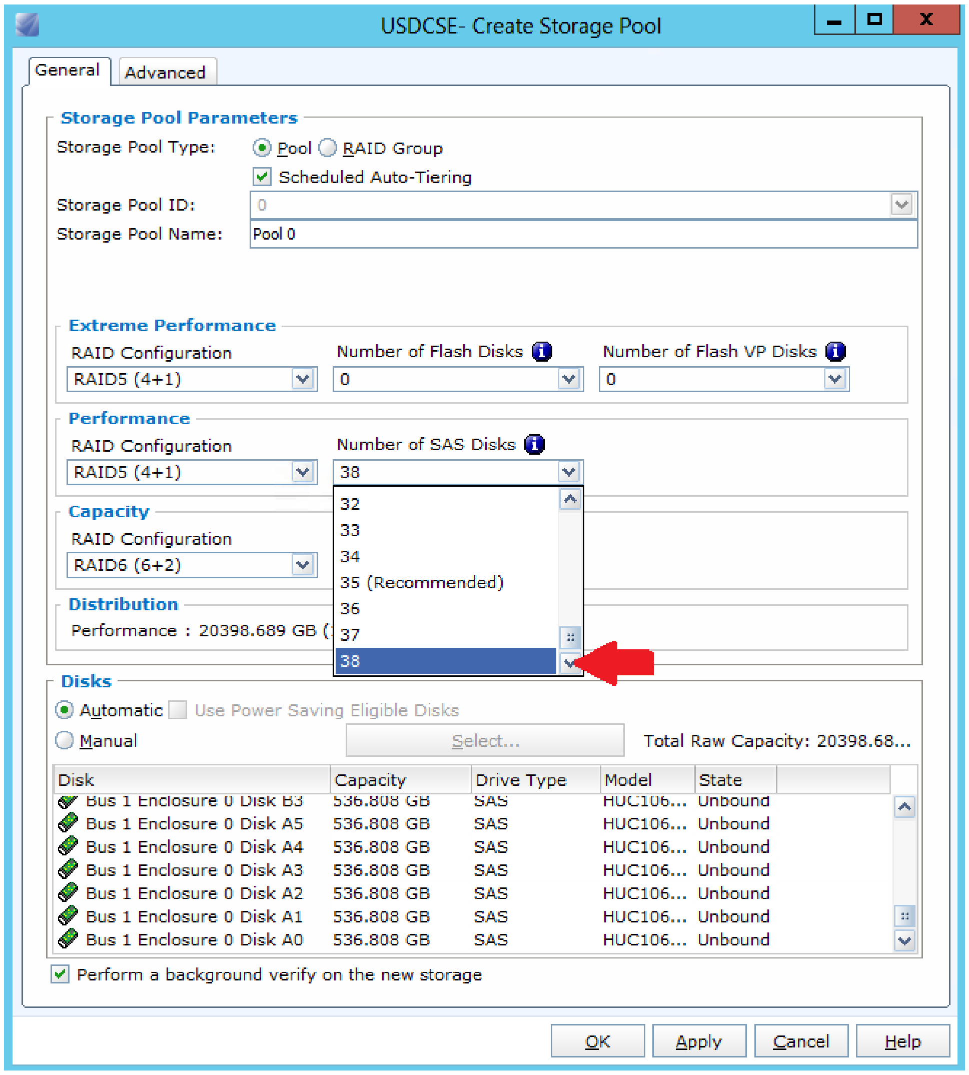
Filter for

!	Disk Type	Total Disks	Hot Spare Policy	Keep Unused per Policy	Unused Disks for Hot Spares
✓	SAS Flash VP(366.797 GB)	5	Recommended	1 per 30 (1 disks)	5
✓	SAS(536.808 GB)	42	Recommended	1 per 30 (2 disks)	40
✓	SAS(268.403 GB)	10	No Hot Spares	1 per (0 disks)	0
✓	SATA Flash(91.727 GB)	3	Recommended	1 per 30 (1 disks)	3
✓	NL SAS(2751.527 GB)	9	Recommended	1 per 30 (1 disks)	9

Figure 37. Hot spare policy

When creating a storage pool out of SAS 600GB (See Figure 38), the Unisphere *Create Storage Pool* wizard would only show 38 available drives, as there are 40 unbound available drives and 2 drives are reserved for hot spares by policy.

Note: In the above example, EMC recommends using 35 drives for a healthy performance tier, not 38 drives. Utilizing 38 drives would force a non-standard sized private RAID group to be created within the Pool.


USDCSE- Create Storage Pool

Storage Pool Parameters

Storage Pool Type: [✓ Pool] RAID Group

[✓ Scheduled Auto-Tiering]

Storage Pool ID: 0

Storage Pool Name: Pool 0

Extreme Performance

RAID Configuration	Number of Flash Disks	Number of Flash VP Disks
RAID5 (4+1)	0	0

Performance

RAID Configuration	Number of SAS Disks
RAID5 (4+1)	38

Capacity

RAID Configuration: RAID6 (6+2)

Distribution

Performance : 20398.689 GB (

Disks

[✓ Automatic] Use Power Saving Eligible Disks

Manual

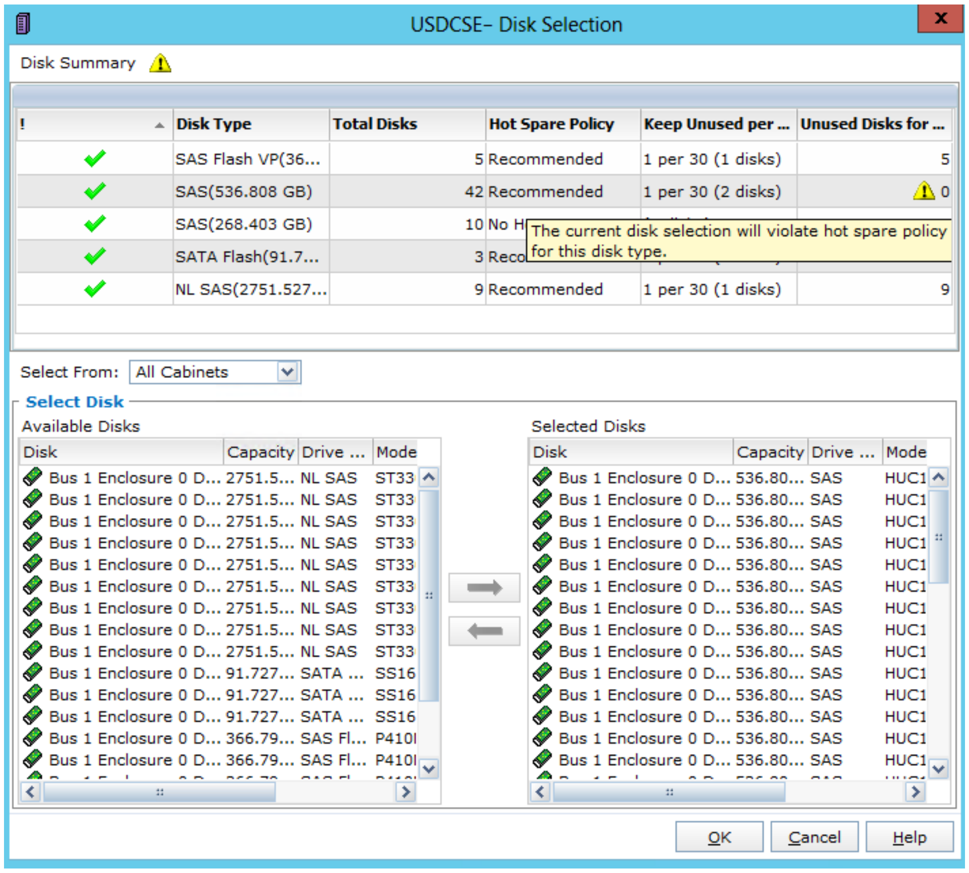
Total Raw Capacity: 20398.68...

Disk	Capacity	Drive Type	Model	State
Bus 1 Enclosure 0 Disk B3	536.808 GB	SAS	HUC106...	Unbound
Bus 1 Enclosure 0 Disk A5	536.808 GB	SAS	HUC106...	Unbound
Bus 1 Enclosure 0 Disk A4	536.808 GB	SAS	HUC106...	Unbound
Bus 1 Enclosure 0 Disk A3	536.808 GB	SAS	HUC106...	Unbound
Bus 1 Enclosure 0 Disk A2	536.808 GB	SAS	HUC106...	Unbound
Bus 1 Enclosure 0 Disk A1	536.808 GB	SAS	HUC106...	Unbound
Bus 1 Enclosure 0 Disk A0	536.808 GB	SAS	HUC106...	Unbound

[✓ Perform a background verify on the new storage]

Figure 38. Create storage pool

If a user switches to *Manual Disk* selection, the array will allow all 40 drives to be selected. However, a Hot Spare Policy violation will occur and an alert will be displayed, as shown in Figure 39.


USDCSE– Disk Selection

Disk Summary ⚠

!	Disk Type	Total Disks	Hot Spare Policy	Keep Unused per ...	Unused Disks for ...
✓	SAS Flash VP(36...	5	Recommended	1 per 30 (1 disks)	5
✓	SAS(536.808 GB)	42	Recommended	1 per 30 (2 disks)	⚠ 0
✓	SAS(268.403 GB)	10	No H		
✓	SATA Flash(91.7...	3	Reco		
✓	NL SAS(2751.527...	9	Recommended	1 per 30 (1 disks)	9

The current disk selection will violate hot spare policy for this disk type.

Select From: All Cabinets

Select Disk

Available Disks

Disk	Capacity	Drive ...	Mode
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	2751.5...	NL SAS	ST33
Bus 1 Enclosure 0 D...	91.727...	SATA ...	SS16
Bus 1 Enclosure 0 D...	91.727...	SATA ...	SS16
Bus 1 Enclosure 0 D...	91.727...	SATA ...	SS16
Bus 1 Enclosure 0 D...	366.79...	SAS Fl...	P410I
Bus 1 Enclosure 0 D...	366.79...	SAS Fl...	P410I

Selected Disks

Disk	Capacity	Drive ...	Mode
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1
Bus 1 Enclosure 0 D...	536.80...	SAS	HUC1

OK Cancel Help

Figure 39. Hot spare policy violation

Drive Sizing

Just like FLARE, Multicore RAID formats drives before use. The formatting allocates end-user usable space, and some *Virtual Drive* metadata. Placing drives into a pool or a RAID group also causes some capacity to be placed aside for the RAID metadata. LUNs have some metadata overhead. Refer to Figure 40.

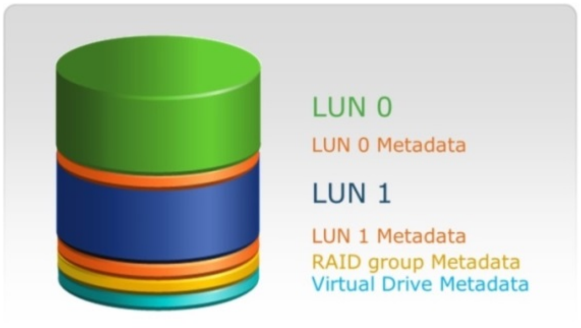


Figure 40. Multicore RAID drive map

Raw drive sizes are the same for both the VNX and the VNX2 series arrays. RAID overhead (including virtual drive metadata overhead) is smaller in Multicore RAID than the RAID overhead in FLARE.

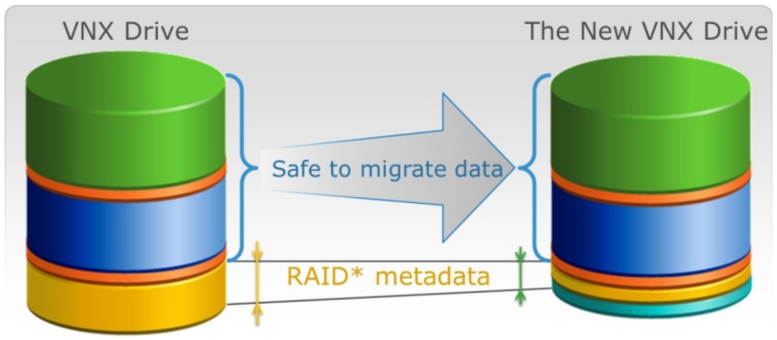


Figure 41. MCx vs. FLARE overhead

The difference is small, but is still positive. For example, an MCx RAID5 4+1 group with 2.5 inch SAS 600GB drives has 5.6GB more usable capacity than a FLARE RAID group with the same drives. Therefore, data migrated from older VNX systems to the VNX2 systems will fit in similarly configured RAID groups or Pools. The reverse is not true: drive space constraints must be considered when planning a migration from the VNX2 back to the older models.

A related subject is drive compatibility between the VNX2 systems and their predecessors. MCx does not recognize the drives that use previous VNX model firmware. Contact a local EMC representative about drive compatibility.

Warning: Do not move a drive from a FLARE array into a VNX2 array. The drive will not be recognized, the slot will continue be marked as 'empty', and the fault light will be enabled on the drive.

RAID6 Parallel Rebuild

Multicore RAID introduces another technology: *Parallel Rebuild*. When a RAID6 group is missing two members, it rebuilds the spares in parallel, unlike FLARE that rebuilds them sequentially.

The following is a sample Parallel Rebuild process, as shown in Figure 42:

1. The first disk in RAID6 fails. The array starts a 5-minute timer. After it expires, a suitable spare is found, and the rebuild process begins. The spare is built from the top.
2. After a spare is 30 percent built, a second drive fails. The array waits another five minutes. The first spare continues rebuilding, and reaches 50 percent.
3. After 5 minutes, a suitable spare is found and the second drive rebuild process begins, in this case at the same block that is rebuilt on the first spare, e.g., starting from a 50 percent mark. Both spares continue to rebuild in parallel.
4. First spare is 100 percent done, and is now a full RAID member. The second spare wraps around, and starts rebuilding from the top, until it reaches the 50 percent mark where the rebuild started.

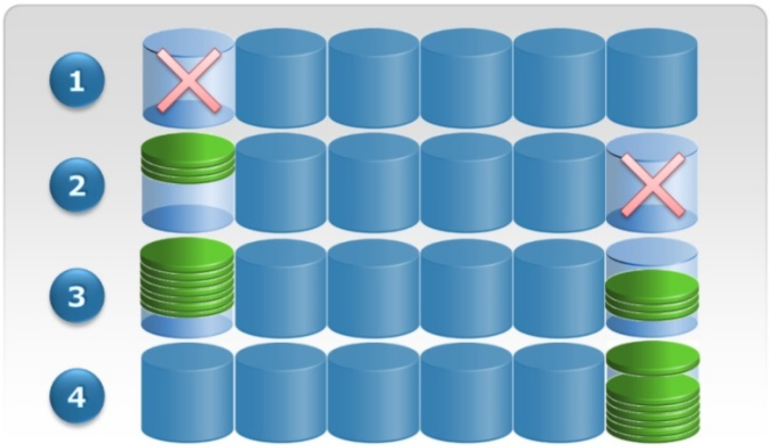


Figure 42. RAID6 parallel rebuild

Drive Zeroing

MCx changes how VNX2 arrays treat new disk drives. Multicore RAID always zeroes new and unknown drives inserted into the system before using them. If a drive is moved between compatible[9] arrays, it is always zeroed first. Drive zeroing is a background operation, but it may affect performance. Drive zeroing cannot be disabled.

Note: New arrays may appear to have slower performance due to drive zeroing; check the logs to determine if drive zeroing is occurring.

The following CLI command shows drive zeroing completion messages:

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0 getlog
| find "0_1_7"

06/20/2013 XX:YY:ZZ N/A  (71680108)Zeroing of disk 0_1_7 (obj 121)
started    00 00 04 00 09 00 2c 00 d3 04 00 00 08 01 68 61 08 01 68 61
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 71 68 01 08
sep

06/21/2013 XX:YY:ZZ N/A  (71680109)Zeroing of disk 0_1_7 (obj 121)
finished    00 00 04 00 09 00 2c 00 d3 04 00 00 09 01 68 61 09 01 68 61
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 71 68 01 09
sep
```

CLI 6. Display disk zeroing log entries

Unlike FLARE, MCx allows the disks to be used while being zeroed. They can be placed into a pool, RAID group, and additionally, LUNs can be bound and data can be written.

Multicore RAID keeps track of the zeroed regions of the disk. When a Host I/O is requested (read or write), Multicore RAID first checks the zeroing map. Figure 43 shows dark green to represent an already zeroed area, and light green, light orange, and light blue to represent areas that still need zeroing. If the Host I/O is addressing a

[9] Please check array and drive compatibility with EMC E-LAB Interoperability Navigator [3].

region that was already zeroed, the I/O is allowed to proceed. If 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.

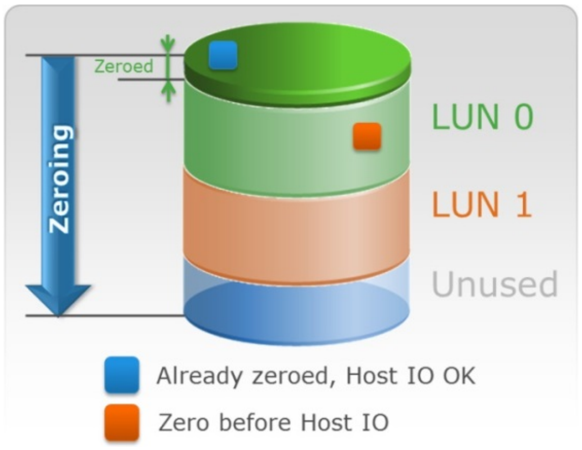


Figure 43. Drive zeroing

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

LUN Zeroing

In some instances, Multicore RAID zeros the LUNs. This happens when a new LUN is overlapping an area that belonged to an already unbound LUN.

The illustration above shows that the drive hosted two LUNs that were already unbound. When a new LUN is bound, Multicore RAID zeroes it with the same rules that apply to drive zeroing: top to bottom, zero all blocks, do not let Host I/O reach an un-zeroed block.

Both LUN and Disk zeroing are performed by the same process. If disk zeroing is ongoing, and the array wants to start LUN zeroing, the request is sent to the same driver that continues to zero the disk.

The LUN zeroing process is aware of pre-existing zeroed blocks. If a new LUN is created that overlaps an older unbound LUN (as in Figure 44), but the unbound areas are unused and still zeroed, the LUN zeroing process simply runs faster—the array does not need to zero the same blocks twice.

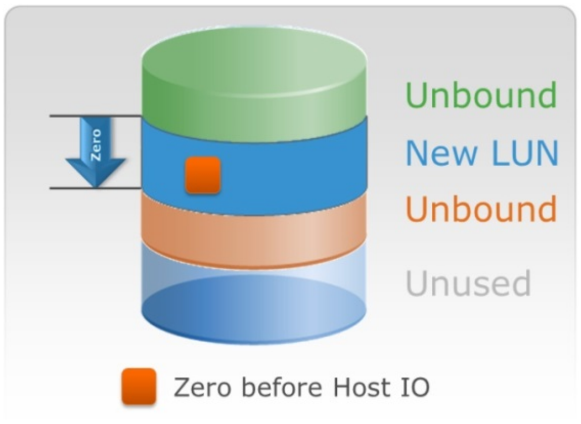


Figure 44. LUN zeroing

Rebuild Logging

When an actively used VNX2 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 RAID group itself is said to be in a *degraded* state when one of its drives is missing. This is true for all redundant[10] RAID types.

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 RAID group disks themselves. Think of RAID metadata as a small hidden LUN at the bottom range of the RAID group.

Figure 45 shows a *Full RAID Stripe* being built in cache (including parity). However, since the RAID group is degraded (member 3 is missing), the data meant for member 3 is discarded in the process of writing the full stripe to the disks (commonly known as *Multicore Cache flush*). The RAID group metadata (shown as RAID Log in the graphic) will be updated with a record of the member 3 blocks that should have been updated as part of a full stripe write.

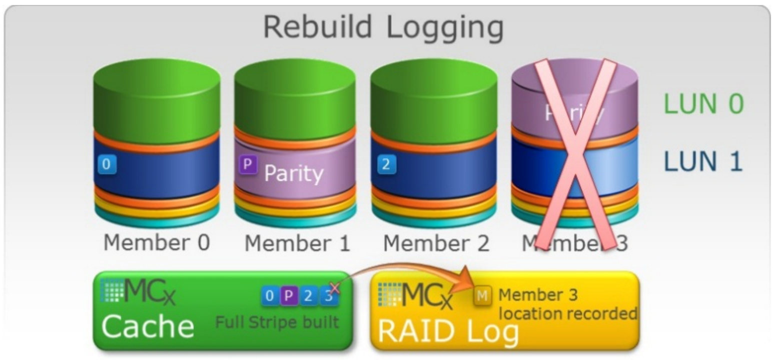


Figure 45. Rebuild logging

[10] VNX2 supported redundant RAID types are: RAID6, RAID5, RAID3, RAID1, and RAID10

For a partial stripe write, the logic remains similar. Parity is calculated based on the full stripe information (missing blocks are read from the disks). However, since member 3 is missing, parity will not be valid after the drive comes back online. Therefore, the log is updated with the location of the parity blocks. After the drive is back online, Multicore RAID simply rebuilds the blocks marked in the log, and turns off rebuild logging.

Rebuild logging is active for the entire duration that a RAID group is in a degraded mode. If an array does not invoke a hot spare after 5 minutes (for example, when unbound compatible drives are not present in the system), rebuild logging continues to run until the RAID group is brought offline or a suitable drive is found.

The log is a bitmap created during the RAID group creation as part of the metadata area. This log defines the size of the RAID group overhead. Therefore, it cannot run out of space. The log is persistent; a rebuild process interrupted by a reboot continues after the power is restored.

Rebuild logging enables VNX2 to avoid full RAID group rebuilds. With it, VNX2 can re-use good data from a drive that was temporarily inactive, significantly reducing the need to rebuild entire drives.

Write Journaling

Data written onto a VNX2 hard drive consists of two parts: the *User Data* and the *Cyclic Redundancy Check* (CRC) [6]. The checksum is a basic protection against media errors. During the read, VNX2 calculates a new CRC and compares it with the one on the disk. 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 VNX2 process called *Background Verify* examines the health of stored data on the disk. 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 disk. It marks a bad media block as *media error*. For more details, refer Background Verify and Sniffer sections.

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, as shown in Figure 46. 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 cache (Multicore Cache).

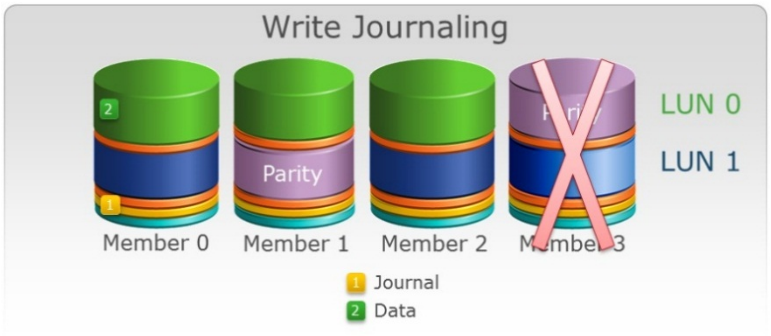


Figure 46. Write Journaling

Uncommitted dirty Multicore Cache pages are protected by the vaulting mechanism (it is discussed in the following System Drives section). After recovering a RAID group from any fault condition (for example, powering on after an unscheduled power off, or 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.

Write journaling is active only when a RAID group is in a degraded state, just like rebuild logging—the two technologies coexist. Write journaling writes the journal to the RAID group metadata area within the destination drive as shown in Figure 40. The journal contains the actual write plus metadata. The 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 disk write is now doubled. However, the performance cost is an acceptable vice given the benefit. Additionally, write journaling replaces an older FLARE technology called *Parity Shedding,* which minimized data loss during RAID group's degraded state. Parity shedding is no longer used because write journaling offers a better overall solution.

System Drives

MCx changes VNX2 system drives requirements. Just like FLARE, the VNX2 system space occupies the first four drives of the first *Disk Array Enclosure* or *Disk Processor Enclosure*, depending on the VNX2 model. However, with MCx, the size of the system area has increased to almost 300GB per drive.

Note: For simplicity, aim to always use 300GB drives for the system drives.

The VNX2 system area includes:

- The VNX2 boot sector and internal OS LUNs
- The Multicore Cache Vault—a special LUN Multicore Cache uses to store dirty cache pages in the event of a power outage
- NAS control LUNs—a set of six LUNs used for the File data

Everything in the system space is hidden from user view.

When the vault is built with SAS 300GB drives, no free VNX2 user space is left. Larger drives leave some of the drive capacity available, as shown in Figure 47.

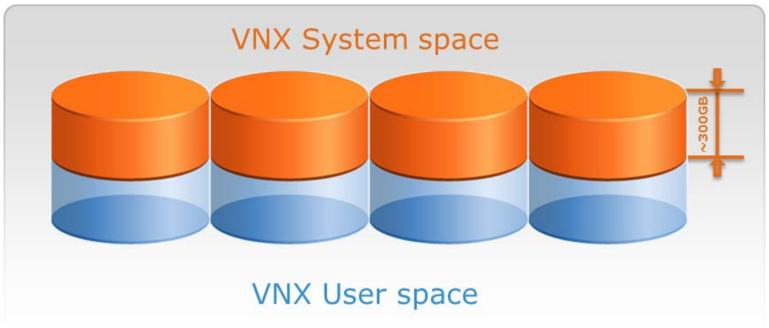


Figure 47. The system drives

The system space does not require a spare because VNX2 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 standard RAID groups, but not Pools. When a vault drive fails, the user space data will be spared onto a suitable spare drive.

If a system drive is moved to another slot, VNX2:

1. Marks the system drive as 'removed'
2. Formats and zeros the former system drive in the new slot
    a. All data (including the private vault system and user spaces) on the former system drive will be removed (zeroed)
    b. If the moved system drive was a member of a RAID group, the user space portion will be 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 VNX2 array:

1. Marks the moved drive as 'removed'
    a. *Starts a 5-minute timer* toward sparing the moved drive to a suitable replacement
2. Formats and zeros the drive in the system slot
    a. *Reserves 300GB system space* of it
    b. Leaves the remaining space unused (if any)

Multicore RAID Summary Examples

For the purpose of the following scenarios, review the 3+1 RAID5 example in Figure 48. When member 3 drive fails, the RAID group becomes degraded, and both rebuild logging and write journaling start. A 5-minute timer activates delaying the sparing process. The performance of the RAID group is slowed down (write journaling overhead).

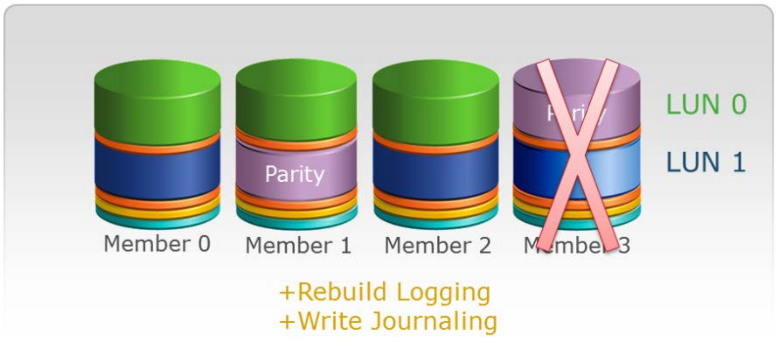


Figure 48. Rebuild Logging and Write Journaling

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 the VNX2 cache (Multicore Cache) and marked as *dirty*. At some point, Multicore Cache will clean the data to the disk. This prompts a read from member 2 disk to calculate parity. Now, parity and host I/O writes are written to the RAID group—original host I/O to RAID member 0 and parity to RAID member 1.

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 3. Write journaling stops, rebuild logging stops, and the RAID group mode is changed back to normal.

Scenario B: Another drive fails

If one more disk fails (for example, member 0), the RAID group becomes *faulted*. A degraded RAID5 group cannot survive two disk failures. RAID6, however, remains in degraded mode, unless three disk members fail. When moving live drives from slot to slot, be aware of the increased failure risk.

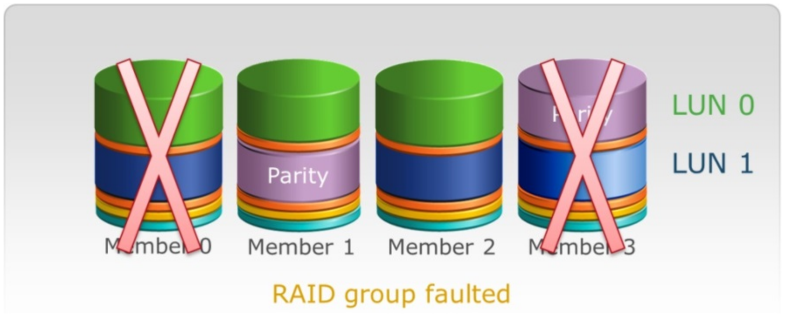


Figure 49. Another drive fails

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. Please refer to Figure 49 above.

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 array is operating normally.

Scenario B1

The moment member 0 drive is removed, the RAID group's state changed to *faulted*, and it stops accepting all I/Os.

The data written onto other drives is intact. The data written onto member drive 0 is protected by write journaling; even though it may have a write hole, the journal will recover that. The data that has not been cleaned from Multicore Cache is still in Cache—the ACK has not been returned, and data continues to be marked as dirty. Multicore Cache will post a warning about the dirty data that cannot be saved to disk due to a disk failure. All I/O activity to the RAID becomes 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 (similar to an FSCK process, if LUNs were file systems). If no contingencies are found, the RAID group is brought back online. Rebuild logging and write journaling are restarted. The dirty cache is flushed.

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

Scenario B2

If drive 0 is faulted and unrecoverable, the data from it is no longer available. There is no way to recover the 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 array can restart this RAID group.

Scenario C: 5-minute interval expires

After the 5-minute interval expires, the array 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 be considered a new drive and 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. VNX2 does not interrupt an ongoing rebuilding of the RAID group.

Summary

MCx is one of the most advanced storage systems stacks available. As described in this paper, it offers substantial innovation 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 4.8TB of data cache.

MCx drives EMCs VNX2 midrange storage experience to new highs, enabling EMC customers to cut storage cost per virtual machine seat up to 300 percent.

Appendix A: Major Differences between MCx and FLARE

This section summarizes the changes from FLARE that MCx brings.

New Drives

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

Usable Space per Drive

Equivalently sized drives have the same usable space on all VNX2 systems (including earlier hardware). RAID group overhead is almost the same with Multicore RAID as it is with FLARE. MCx RAID overhead is a tiny bit smaller.

When sizing VNX2 implementations, it is safe to use older VNX RAID group capacity numbers. The usable capacity remains the same for the same sized drives.

Disk and LUN Zeroing

The array zeros the drives and can zero the LUNs. LUN zeroing is a metadata update, if the LUN was bound on an already zeroed drive. If a LUN is bound on a drive that WAS used either with an unbound LUN, or in a destroyed pool, the LUN area is zeroed after LUN creation.

SCSI Priority

MCx includes a technology that sets priority for SCSI commands for the back-end drives. The presence of the SCSI priority warrants new drive firmware and formatting. EMC previously used this technology in the VMAX arrays, and is now using it on the VNX2.

The first implementation includes Multicore Cache write flushes only.

IO Error Retry

The MCx Drive I/O Error retry is time based, unlike FLARE which determines drive health by counting the number of failed I/Os. With Multicore RAID, if an I/O sent to the drive takes longer than expected, the drive can be marked offline. The RAID group proceeds in degraded mode, until either the drive comes back, or it is spared (after 5 minutes).

Multicore RAID keeps a permanent record of all drive errors. An unstable drive quickly becomes a candidate for a proactive replacement. MCx keeps a record of drives by their serial numbers; the health record is maintained even if a drive is moved to a new slot.

Background Verify

Background Verify is an internal array process that inspects and validates the data and parity of bound LUNs. Its job is to detect and correct RAID errors.

Background Verify is started for a few different reasons:

- User manually initiates a background verify operation to detect and correct errors
    - Use `setsniffer` command to start and `getsniffer` command to report on progress
- System start background verify operation automatically for incomplete write error situations
- System start background verify operation automatically when background zeroing on a LUN is complete

With MCx BV has an ability to back off under high load on the RAID group.

Note: Background Verify works only for Classic LUNs and Private Pool LUNs.

Getsniffer/Setsniffer CLI Notes

The switches `–bvtime` (background verify time), `-snrate` (sniffer rate) and option `[0|1]` are not supported with Multicore RAID.

- `[0|1]` is not supported with this command because it was used to enable a sniff verify, which is now supported by using the new `setsniffverify` command.
- The fields in the background verify report are modified. `Sniffing state`, `Sniffing rate` and `Background verify priority` lines were removed from the output. Those properties are no longer applicable with this command when running with Multicore RAID.
- The `-nonvol` switch is not supported on the `getsniffer` command because the Non-Volatile Recovery Verifies report is no longer applicable.

When using the `setsniffer` command, the `-bv` switch must be specified, in order to initiate a background verify. The command also takes the optional `-cr` switch to clear the reports.

Sniffer

Sniff Verify, or *Sniffer*, is an internal process that detects and corrects potential disk media errors before they become unrecoverable. In Multicore RAID sniff verify operates on the entire disk capacity including bound and unbound areas of the disk.

In FLARE, it only sniffs bound areas of the drive and leaves the unused area unchecked. As a result, some media errors are undetected. In FLARE, the sniff verify operation can be scheduled on a LUN basis and only bound regions of the disk are sniff verified.

Multicore RAID does not allow sniffer scheduling. Multicore RAID added the ability to disable/enable sniff verify at the system level. Enabling sniff verify on an individual disk is not supported. Sniffing is enabled/disabled for all the disks on the array. Sniffer is enabled by default.

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0
setsniffverify

sniffverify is ENABLED
```

CLI 7. Enable sniffer on the array

Proactive Sparing

VNX2 monitors drive health by counting the number of errors found. The errors are discovered by 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 engine starts a sparing process without incurring a 5-minute waiting period. Drives do not increment error count forever. The error count is decremented by one after 18,000 good I/Os to the drive.

Features not available in the VNX2 Series

- RAID Group Defragmentation
- RAID Group Expansion

Appendix B: Unisphere Analyzer

While comparing FLARE and MCx, a Multicore Cache metric of Cache Dirty Pages was used. In Multicore Cache this metric is counted in MB used.

Below are the actual graphs made with data taken from Unisphere Analyzer:

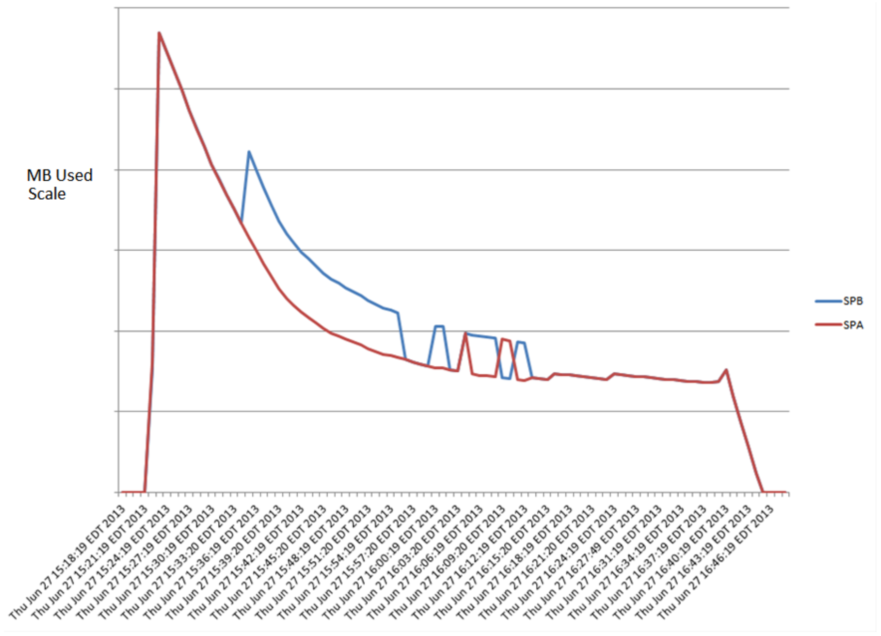


Figure 50. The VNX2 actual NAR data

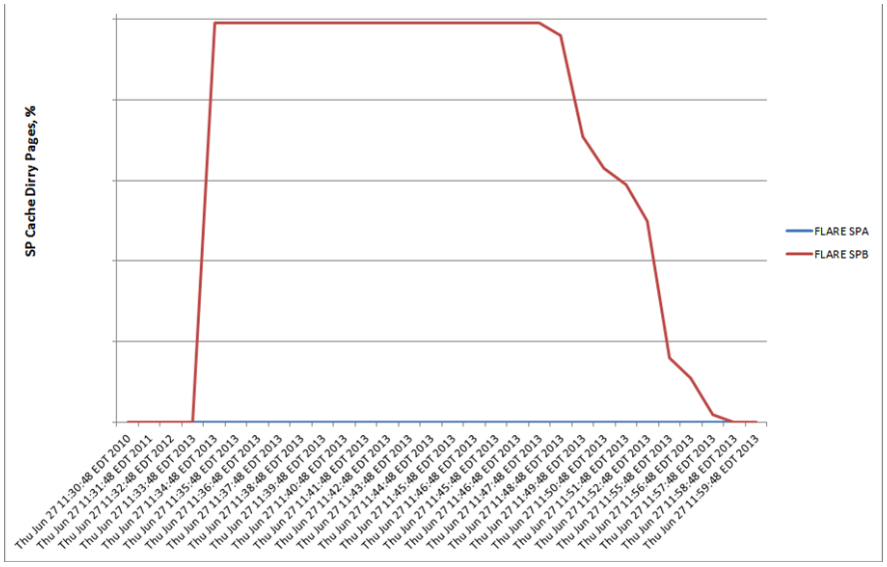


Figure 51. VNX OE 32 actual NAR data

Test Setup Notes

The test was conducted on the following arrays:

- VNX7600 array, running VNX2 OE 33 (beta build 687)
- VNX5500 array running VNX OE 32 (patch 201)
    - Cache thresholds were set at 60%/80% (system default)
    - Write cache was set to be 3915MB
    - Read cache was set to be 435MB

A single 4+1 RAID5 RAID group was created out of five 300GB SAS 10K drives. Then a single 100GB Classic LUN was created and presented to the Windows host. A single SAN switch was used, and only two paths were zoned, one path per VNX SP. The size of the LUN was specifically chosen to be large to ensure that the workload on either system would overwhelm the cache. PowerPath 5.7 was installed on the Windows host.

The I/O workload generation program (Iometer) was used to generate the load. Iometer was configured with a single worker, and 16 outstanding IOs per target. The Access Specification was configured to be a 100 percent random 8KB unaligned write.

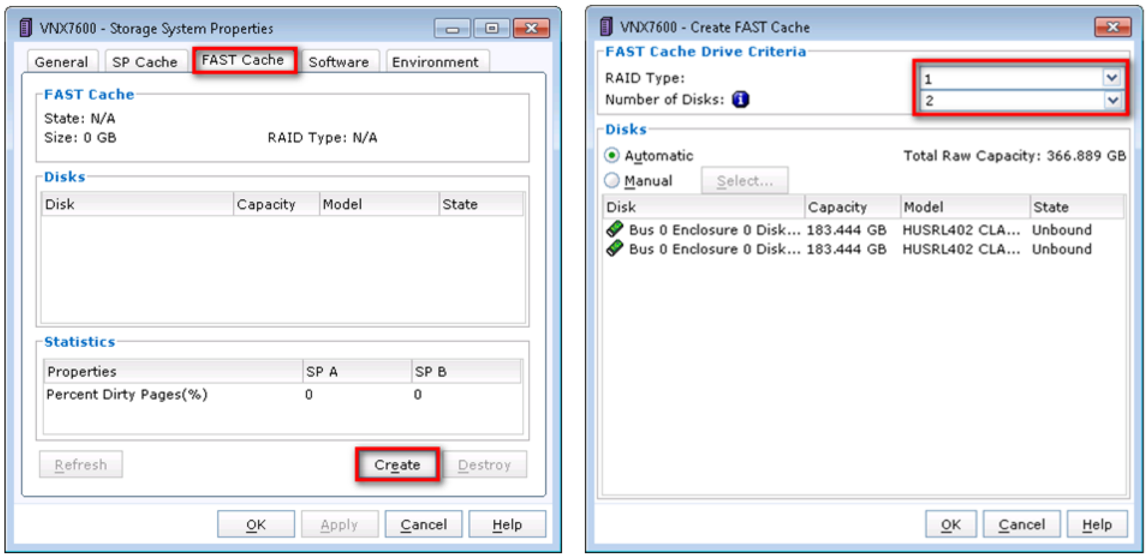
The FLARE test was shorter than MCx in order to allow Multicore Cache dirty pages to plateau.

Appendix C: Multicore Cache Management

Management

To enable MCx FAST Cache in Unisphere, navigate to the System tab, and then click on the System Management › Manage Cache link on the right side of the screen. Click the FAST Cache tab in the Storage System Properties window.

If FAST Cache is not created, the Create button is enabled. Once pressed, it opens the Create FAST Cache window, providing a form to select the flash disks for FAST Cache, as shown in Figure 52 below.



FAST Cache

State: N/A

Size: 0 GB RAID Type: N/A

Disks

Disk	Capacity	Model	State

Statistics

Properties	SP A	SP B
Percent Dirty Pages(%)	0	0

FAST Cache Drive Criteria

RAID Type: 1

Number of Disks: 2

Disks

[✓ Automatic] Manual

Total Raw Capacity: 366.889 GB

Disk	Capacity	Model	State
Bus 0 Enclosure 0 Disk...	183.444 GB	HUSRL402 CLA...	Unbound
Bus 0 Enclosure 0 Disk...	183.444 GB	HUSRL402 CLA...	Unbound

Figure 52. Create MCx FAST Cache

In the CLI, the same process is done by using the following command:

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0 cache
–fast -create 0_0_23 0_0_24

Configuring FAST Cache may take up to several hours to complete. It
will also reduce the amount of Storage System memory available for use
by SP Cache. Do you want to continue? (y/n) y
```

CLI 8. Create FAST Cache

Destruction of FAST Cache

MCx FAST Cache can be destroyed online. When ordered to be destroyed, FAST Cache immediately changes its state to *Disabling*, as shown in CLI 10. Show FAST Cache information. It stops all new data copies, and starts the flushing process, which expunges all pages from FAST Cache flash storage. While the flushing process is running, Multicore FAST Cache still services hosts I/O for the pages that remain in cache.

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

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0 cache
–fast -destroy

To destroy FAST Cache, the Storage System may flush all data to disk.
This operation may be time consuming and may impact system performance.
You can monitor the progress of this operation by using the following
command: cache -fast -info. Do you want to continue? (y/n) y
```

CLI 9. Destroy FAST Cache

```
C:> naviseccli -h VNX-SPA –User user –Password password –Scope 0 cache
–fast –info

Disks:
Bus 0 Enclosure 0 Disk 21
Bus 0 Enclosure 0 Disk 24
Bus 0 Enclosure 0 Disk 23
Bus 0 Enclosure 0 Disk 22
Mode:  Read/Write
Raid Type:  r_1
Size (GB):  183
State:  Disabling
Current Operation:  Destroying
Current Operation Status:  Running
Current Operation Percent Completed:  0
Percentage Dirty SPA:  0
MBs Flushed SPA:  1585
Percentage Dirty SPB:  70
MBs Flushed SPB:  93345
```

CLI 10. Show FAST Cache information

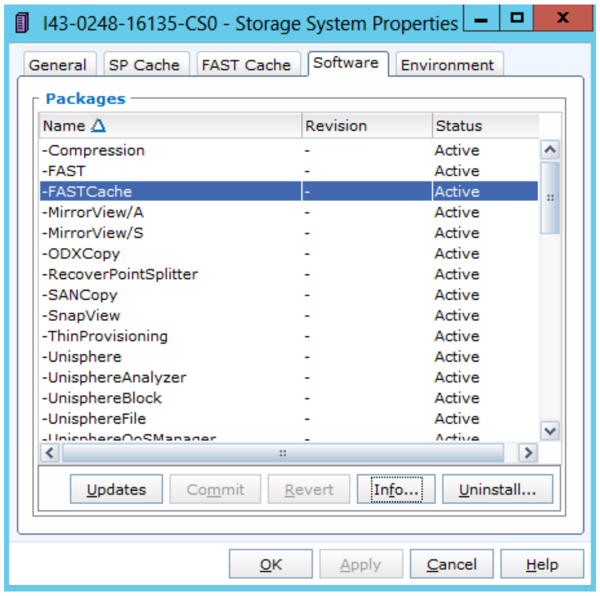
FAST Cache Enabler

Multicore FAST Cache is an optional member of the MCx stack. A special *FAST Cache enabler* must be installed on the system to permit FAST Cache functionality. Installing a FAST Cache enabler does not require rebooting a Storage Processor. Licensing is outside the scope of this document; contact an EMC representative with licensing questions.

Several SAS Flash or SAS Flash 2 drives need to be grouped in mirrored pairs to activate FAST Cache. The maximum count of the mirrored pairs varies depending on the model of the VNX array and the available flash drives. See Multicore FAST Cache Configuration Options section for details.

The presence of the FAST Cache enabler changes the defaults for newly created LUNs. If the enabler is installed, all new Classic LUNs and Pools[11] will have 'FAST Cache' option enabled by default. To check whether the enabler is installed, in Unisphere navigate to the System tab, and then click the System Management › System Properties link on the right side of the screen, as shown in Figure 53.

[11] Pool LUNs do not have FAST Cache option, as it is controlled on the Pool level.


I43-0248-16135-CS0 - Storage System Properties

Name	Revision	Status
-Compression	-	Active
-FAST	-	Active
-FASTCache	-	Active
-MirrorView/A	-	Active
-MirrorView/S	-	Active
-ODXCopy	-	Active
-RecoverPointSplitter	-	Active
-SANCopy	-	Active
-SnapView	-	Active
-ThinProvisioning	-	Active
-Unisphere	-	Active
-UnisphereAnalyzer	-	Active
-UnisphereBlock	-	Active
-UnisphereFile	-	Active
-UnisphereQoSManager	-	Active

Figure 53. An installed FAST Cache enabler

All LUNs created before the enabler is installed have the 'FAST Cache' option disabled, even though Unisphere does not show the option until the enabler is present. To enable FAST Cache usage for the LUNs and/or pools created prior to FAST Cache enabler installed, manually check the option.

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 others.

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, dirty pages are written back to the user LUNs. New hot pages from any user LUNs will be promoted to the other remaining[12] 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 are stored on that device.

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

Power Faults Table

All power faults in Table 1 below are longer than the 15-second power glitch ride-through.

[12] This will not be true if FAST Cache only has a single pair of assigned flash drives, or if all pairs are degraded.

Table 1. Multicore Cache Fault Handling

Use Case	**Description**	**Multicore Cache State**
Single AC power failure, peer SP still running	AC power is removed from one SPS in a system with multiple SPS (1 or 2 per SP)	Enabled
Single SPS power failure, peer SP not running	AC power is removed from one SPS in a system with multiple SPS (1 or 2 per SP) and the peer SP is not present.	Vault dump
AC power failure to all SPS	AC power has been removed from all SPS.	Vault dump
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 SP and power supplies.	Vault dump
DAE 0 (0_0) over temperature	An over temperature condition is reported by one or more power supplies in the first drive enclosure	Vault dump
Single SPE/DPE power supply fault	A single power supply is reported as faulted in the SPE.	Enabled
Single 60-drive DAE power supply fault	A single power supply in the first DAE is reported as faulted and first enclosure is a 60-drive enclosure	Vault dump
Multiple power supply faults	Multiple power supplies are reported as faulted.	Enabled
Single fan fault	A single fan has faulted.	Enabled
Multiple fan faults	Multiple fans are reported as faulted.	Enabled
Degraded vault	A single system drive is offline or has been removed.	Enabled
Faulted vault	Multiple system drives are	Disabled

Use Case	Description	Multicore Cache State
	offline or have been removed.	
NDU without power supply firmware upgrade	Array software is being changed	Enabled
NDU with power supply firmware upgrade	Array software and the power supply firmware is being changed	Disabled
Single SP fault	A single SP in a dual SP system is removed or rebooted or panics or has a hardware fault	Enabled
Single SPS communication fault	A SP is unable to communicate with a single SPS. Other SPS is OK.	Enabled
Multiple SPS communication faults	Neither SP is able to communicate with the SPS(s) connected to that SP.	Disabled
Single SPS cabling fault	An SPS is incorrectly cabled	Enabled
Multiple SPS cabling fault	Multiple SPS are incorrectly cabled on both SP	Disabled

Appendix D: VNX2 Flash Drive Technology

The following table outlines the underlying drive technologies of the Flash drives used in VNX2 storage systems. The table shows the shows the Unisphere label for each Flash drive, the drive sizes for each type of Flash drive, and the underlying technology for each of the types of drives.

Unisphere Label	Drive Size	Drive Technology
SAS Flash	100 GB	Single-Level Cell (SLC) / Enterprise Multi-Level Cell (eMLC)
	200 GB	
SAS Flash 2	100 GB	Enterprise Multi-Level Cell (eMLC)
	200 GB	
	400 GB	
	800 GB	
	1.6 TB	
SAS Flash 3	400 GB	Triple-Level Cell (TLC) / Equivalent
	800 GB	
	1.6 TB	
	3.2 TB	

References

[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

[2] SCSI Block Commands - 3 (SBC-3), T10 working draft specification, http://www.t10.org/cgi-bin/ac.pl?t=f&f=sbc3r32.pdf

[3] EMC E-Lab Interoperability Navigator, An EMC website, https://elabnavigator.emc.com

[4] Applied Best Practices Guide: EMC VNX Unified Best Practices for Performance, EMC Support website, https://support.emc.com/docu42660_Applied_Best_Practices_Guide:_EMC_VNX_Unified_Best_Practices_for_Performance.pdf

[5] Haswell chip, Intel Corporation, http://www.intel.com/content/www/us/en/secure/intelligent-systems/privileged/core-qm87-hm86-chipsets-is-guide.html?wapkw=haswell

[6] 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

[7] EMC FAST VP white paper, EMC Support website, https://support.emc.com/docu32691_White_Paper:_EMC_FAST_VP.pdf?language=en_US