
EMC CLARiiON Fibre Channel Storage Fundamentals

Abstract

This white paper explains the effects of host operating system parameters, storage controllers, RAID type, and disk types on performance and redundancy in a Fibre Channel storage system.

Published 10/31/2003

Copyright © 2003 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, Symmetrix, AutoIS, Celerra, CLARiiON, CLARAlert, DG, E-Infostructure, HighRoad, Navisphere, PowerPath, ResourcePak, SnapView/IP, SRDF, VisualSAN, where information lives, WideSky, and The EMC Effect are registered trademarks and EMC Automated Networked Storage, EMC ControlCenter, EMC Developers Program, EMC Enterprise Storage, EMC Enterprise Storage Network, EMC OnCourse, EMC Proven, EMC Snap, Access Logix, AutoAdvice, Automated Resource Manager, AutoSwap, AVALONidm, C-Clip, CacheStorm, Celerra Replicator, Centera, CentraStar, CLARevent, Connectrix, CopyCross, CopyPoint, CrosStor, Direct Matrix, Direct Matrix Architecture, EDM, E-Lab, Enginuity, FarPoint, FLARE, GeoSpan, InfoMover, MirrorView, NetWin, OnAlert, OpenScale, Powerlink, PowerVolume, RepliCare, SafeLine, SAN Architect, SAN Copy, SAN Manager, SDMS, SnapSure, SnapView, StorageScope, SupportMate, SymmAPI, SymmEnabler, Symmetrix DMX, TimeFinder, Universal Data Tone, and VisualSRM are trademarks of EMC Corporation. All other trademarks used herein are the property of their respective owners.

Part Number H1049

Table of Contents

Executive Summary	5
Intended Audience	5
Introduction	5
Considerations for Performance	6
Visualizing the Performance Problem	6
Defining Performance	6
Characterizing I/O	7
Application Design	8
Why Tuning Host Software Is Important	9
Application and I/O Characterization	9
Host File System Impact	11
File System Buffering	11
Minimum I/O Size: The File System Request Size	12
Maximum I/O Size	12
File System Coalescing	12
File System Fragmentation	12
File System Alignment	12
Raw Partitions Versus File Systems	13
Volume Managers	14
Host-Based Striping (Plaids) and Channel Throughput	14
Host Virtual Memory	16
Host HBA Effects	17
Failover and Multipathing	18
Storage Controller Effects	18
CLARiiON Storage Processors	19
RAID Levels	19
MetaLUNs	21
The CLARiiON Cache	22
Cache Size and Performance	22
Write Cache Mirroring	23
Activating the Write Cache	23
Cache Settings	24
The Back End	27
LUN Distribution	27
Maximum Spindle Counts	28
Minimizing Disk Contention	28
Stripes and the Stripe Element Size	29
Increasing the Spindle Count	29
How Many Disks to Use	29
Disk Types	30
Considerations for Reliability and Redundancy	30
Highly Available Attach	31
RAID Level Considerations	31
RAID 5	31
RAID 1/0	32
Binding Disk Groups across Disk Shelves	32

Summary	32
Glossary	33
Appendix A: RAID 5	36
The Parity Write Penalty	36
CLARiiON RAID 5 Optimizations	37

Executive Summary

This white paper discusses the primary factors that impact storage-system performance and **reliability** for CLARiiON® storage systems. These storage systems provide block-level storage by means of direct Fibre Channel attach. The focus is on performance, but availability concerns—particularly the relative advantages of RAID types used—are discussed as well.

Performance concepts are explained in context throughout the paper.

Intended Audience

This white paper is intended for EMC customers, EMC systems engineers, EMC partners, members of the EMC and partners professional services community, and anyone who requires a good working knowledge of how various system components affect performance and reliability in a Fibre Channel storage system. Readers of this document are required to understand the basics of mirrored and parity RAID, and should have familiarity with storage terms and operating system functions.

Introduction

This paper outlines the factors that affect the performance and availability of CLARiiON Fibre Channel, block-level storage.

The paper takes a step-by-step approach to explaining performance—how it is measured and how it is impacted. The various system components that affect performance are addressed in succession—from the application, to the host, to the storage system itself. The characteristics of each component are explained.

This document covers concepts that deliver redundancy and reliability as well.

A glossary is located at the end of this paper. The first time glossary terms appear in the paper they are shown in **boldface** and linked to the definition in the glossary.

Considerations for Performance

How important is performance tuning? The typical answer you receive from performance experts is: “it depends.” This section explains what the dependencies are and how they affect different applications.

Visualizing the Performance Problem

The most challenging part of analyzing a system for its performance capability is the presence of many layers between the user (whose complaints guide the system administrator) and the storage system itself. To assist with this, a layer diagram (Figure 1) is used throughout the paper. The layer diagram displays the storage *performance stack*. This is a conceptual **device** used to address the implications of various layers of software on performance. (It does not represent an actual stack in the software engineering sense.)

Like any stack, each layer depends on the ones below it, and each layer affects performance in some way. The discussion of the performance stack starts at the top—at the host—and works down.

This may appear nonintuitive, but the storage system cannot be optimized until the application and host behavior have been characterized.

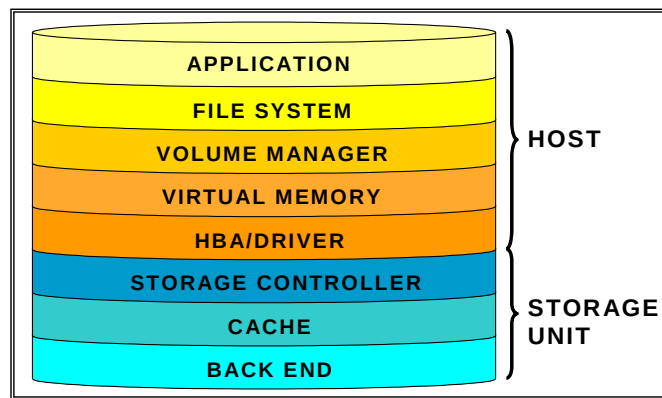


Figure 1. Performance Stack

Defining Performance

You can measure performance in several ways. The most common definitions used to measure performance are:

- **Throughput** — The number of individual I/Os the storage system can process over time. It is measured in I/Os per second (IOPS).
- **Bandwidth** — The amount of data the storage system can process over time. It is measured in megabytes per second (MB/s).
- **Response time** — The interval of time between submitting a request and receiving a response.

Of these, response time is the measurement most visible to users. Response time is sometimes referred to as **latency**. However, this is actually a misnomer; latency is specific to disk-drive response time.

All layers of the performance stack affect response time, so it provides a good, comprehensive measurement. More importantly, however, *response time is what users complain about*. A typical IS department manager responds to complaints when applications work too slowly.

Response time, being a function of I/O load, is used along with other measurements, such as bandwidth, to plot **saturation**. As the load on a system is increased, **throughput** and bandwidth increase. However, when a system approaches the limits at which it can operate, response times increase exponentially. Figure 2

shows a chart from an **online transaction-processing (OLTP)** performance profile. Response time is plotted for two storage systems as the load is increased.

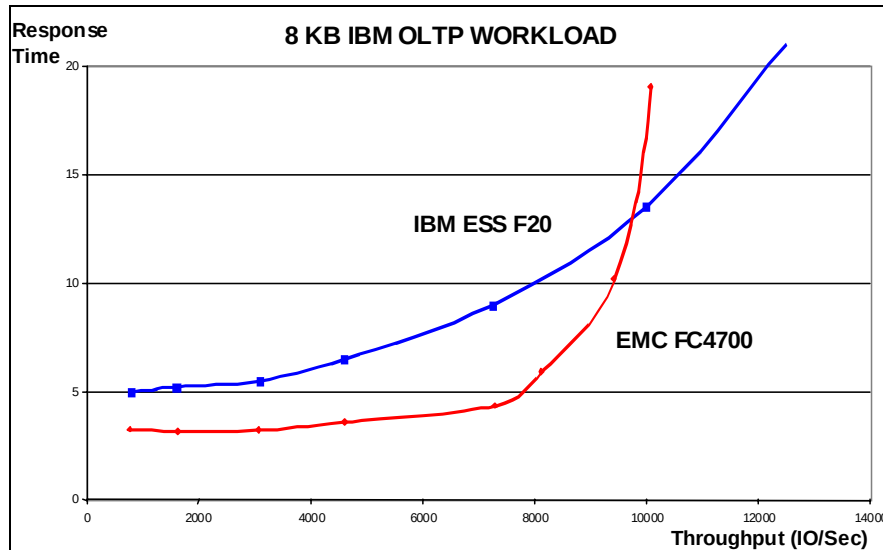


Figure 2. Dual-Host OLTP Simulation

Note that the FC4700 reaches a point where its response time increases dramatically. This is saturation. The ESS is still increasing gradually, which shows that it has an overall greater OLTP capability. On the other hand, it is interesting to note that until the FC4700 saturates, the ESS is always slower—servicing I/O with a longer response time—than the FC4700. Knowing the response-time characteristics of a system is as important as knowing the maximum I/O capacity.

Characterizing I/O

It is important to recognize how and why I/O is characterized.

Random Versus Sequential

I/O can be characterized as **random** or **sequential**. *Random* refers to successive reads or writes from noncontiguous addresses—accesses spread across the drives. *Sequential* refers to successive reads or writes that are physically contiguous—one after the other across the drive.

Since data is stored on rotating disks, *the system gets much better performance from sequential I/O* because the next location in the dataset is the next location on the disk, and the disk head does not have to move to access the next location. Thus, latency at the drive level is reduced when contiguous blocks are accessed. Storage systems leverage this characteristic by reading ahead many blocks when sequential access is detected, a process known as **prefetch**. The data is read and cached for fast access by the host.

Sequential writes are also effective and, again, caching allows storage systems to optimize sequentiality. In addition, sequential writes by the host allow a different form of optimization: coalescing. Many smaller writes can be coalesced into fewer large transfers to the disk drives. Sequential writes also enable other optimizations.¹

Random I/O requires the disk to seek for subsequent requests. Disk head movement is a relatively slow mechanical process. Also, it is very difficult to write optimizations for random access. For example, to pre-fetch tracks for random reads, runtime **logical block address (LBA)** tracking and statistical analysis is required. These latter features are found in high-end storage systems.

¹ Refer to *Appendix A: RAID 5* for details on parity stripe optimization.

Reads and Writes

Another aspect of I/O is whether it is a *read* or a *write* transaction. This is chiefly a matter of storage processor (SP) caching. Caching is how the CLARiiON storage system provides much of its performance potential. Reads and writes must be handled differently in a caching system.

Performance of reads and writes, and our ability to cache, is closely associated with the randomness of the data. Table 1 summarizes how reads and writes interact with the cache.

Table 1. Read/Write Interactions with Cache

I/O Type	Read	Write
Random	Hard to effectively cache (prefetch not typically <i>hit</i>), requires multiple fast disks for good performance.	Caching is effective, resulting in better-than-disk speed.
Sequential	Caching is extremely effective as prefetched data is <i>hit</i> in high percentages; reads are done at cache speeds.	Caching is effective; cache is flushed quickly as entire disk stripe elements can be written; CLARiiON optimizes RAID 5 so stripes are written with no penalty. ²

I/O Request Size

Finally, the request size of the I/O is important. Larger I/Os take longer to transmit over the Fibre Channel, and longer to mirror in the write cache. However, some of the overhead to execute an I/O is fixed, so if data exists in large chunks, it is more efficient to transmit larger blocks; a host can move more data faster by using larger I/Os than smaller ones. The response time of each large transfer will be longer than the response times for a single smaller transfer, but the combined service times of many smaller transactions will be greater than a single transaction that contains the same amount of data.

On the other hand, if the application accesses only very small chunks of data, this must be done very quickly. The CLARiiON storage system achieves its best rate of IOPS with small I/O sizes: 512 bytes to 8 KB.

Bursts and Busy Periods

Burstiness is not so much a characteristic of I/O as it is of the system as a whole. Identifying busy periods—or cases where bursts of activity are experienced—is important when designing storage. Systems must be designed to have reserve capacity so that bursts do not result in unacceptable service times.

Application Design

This section describes the application layer of the performance stack—the end user of the storage. Without an application, there would be no data to store. The application introduces the need for storage. The application design determines much of the behavior of the system. Some application characteristics discussed in this section are:

- The efficiency of the application
- The character of the I/O
- Patterns in data access
- **Buffering**

² Refer to *Appendix A: RAID 5* for details on RAID 5 optimizations.

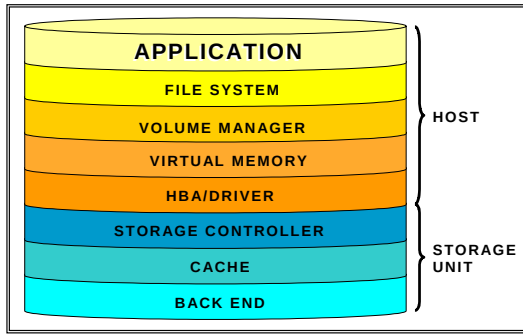


Figure 3. Application Layer of the Performance Stack

Why Tuning Host Software Is Important

All of the previously mentioned application characteristics affect response time. In the performance stack (Figure 3), note that at least half of the factors affecting response time are due to the host (and software running on the host).

A tenet of software design is: even very fast hardware will not overcome bad software design. If the application is poorly designed, or not tuned, tuning the storage system will have little or no effect.

Consider this example: a small chain of discount fashion stores uses computers as registers, and every night they download the day's receipts from all the stores and compute the accounting reports. It takes eight hours for six stores. After doubling the hardware speed, the job takes only six hours. A comparable system, used by a similar chain, completes the job in 45 minutes. The difference is not hardware, which was comparable—it is a better database design.

Application performance can suffer from many causes: bad sort algorithms, repeated requests for the same data, or the use of inefficient database-access routines. Even good database systems can perform poorly due to bad implementation.

Tuning of host buffers, database global memory (such as Oracle System Global Area), and defragmentation of swap files all will help keep the host utilizing the storage system effectively.

Application and I/O Characterization

The application is important in that it affects the character of the I/O. As introduced in the *Defining Performance* section on page 6, the characteristics of I/O are:

- Sequential or random
- Reads versus writes
- I/O size
- Patterns in data access: bursty or steady

Although the I/O profile is critical in performance tuning, it is often unknown. This section characterizes some common activities and their typical I/O profiles.

Sequential or Random I/O

Examples of sequential I/O include any type of cumulative addition to tables, such as adding new users, appending real-time data for later analysis, and file backup. Transaction log activity is sequential.

Examples of random updates include client account balance updates, inventory tracking, and accounting.

Reads Versus Writes

Typical mixtures are:

- **Online Transaction Processing (OLTP)** — 67 percent reads and 33 percent writes. No table scans.
- **E-mail** — Very similar to OLTP. Typically less intensive per GB of storage.
- **Decision Support System (DSS)** — Applications: 80 to 90 percent reads to data tables, frequent table scans (sequential reads), and heavy write load on temporary space.
- **Backup** — Don't forget backup operations. As long as the file system is not fragmented, file-based backups will be sequential. Application-based backup processes, such as Microsoft Exchange online backup, are often random.

I/O Size

The I/O size is, for the most part, determined by the application and the framework upon which it is built. For example, the request sizes of the applications from SAP are typically small. But in the underlying database engine—typically Oracle—the I/O request size is controlled by environment variables such as *read_ahead_cluster_size*. This variable would be set up differently for an Oracle instance doing reporting/data mining than for an instance servicing OLTP or SAP applications.

A tool such as Navisphere[®] Analyzer is irreplaceable in determining the size of the I/O that the storage system sees in a complex environment.

Temporal Patterns and Peak Activities

In OLTP applications, transactions that perform batch-type functions (daily receipts or weekly reports) can cause bursts—or **spikes**—of I/O, which consume resources on both the host and the storage system. Client loads to watch for include quarter/year-end closings, meal times, scheduled events, and busy days (Friday/payday/holidays).

Spikes are either localized (to certain file systems) or global. Global resources, such as database buffers and storage cache, must have the reserve capacity to accommodate spikes. The disk group hosting a bursty file system must also have reserve capacity: more drives can handle more I/Os. Drives hosting bursty data should be no more than 70 percent utilized under nonpeak conditions.

Application Buffering and Coalescing

Some applications, such as RDBMSs, **buffer** data using host memory. Buffering is the process of putting data about to be written in a special area or RAM—the buffer—for retention and reuse. Buffering is also used in reads by reading ahead and placing data the application expects to use in its buffers.

The application's creators know best what data—or what type of data—is best to retain in its buffers. Therefore, buffering benefits performance. Contrast application buffering to file system buffering or storage system caching, neither of which make use of application design to decide which data to retain in the buffers or cache.

An RDBMS also **coalesces** writes. Coalescing is the process of gathering multiple smaller I/O requests into one larger request. As pointed out in the *Characterizing I/O* section on page 7, coalescing can hinder or help performance by causing I/O to be large enough to bypass the CLARiiON write cache.

Applications Summary

This section examines typical applications and their characteristics.

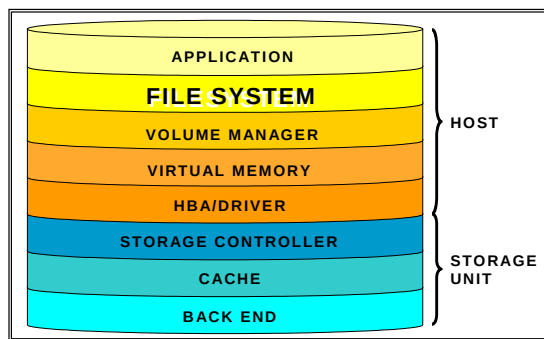
Table 2. Typical Applications and Their Characteristics

Application Processing	Seek Type	I/O Request Sizes	Percent of I/O as Writes
Microsoft Exchange	Very random	4 KB	Moderate-high
Microsoft Exchange Backup	Random	64 KB	Very low
SAP/Oracle applications	Very random	~4 KB (underlying database page size)	Depends on application
RDBMS: Data entry/OLTP	Very random	Database or file system page size	Moderate-high
RDBMS: Reporting/DSS	Some sequential	Record size (8 KB+) Binary data (photos) can be large	Low (mostly reads)
RDBMS: Online (transaction) logs	Sequential	Database or file system page size	Moderate (complete replacement at interval)
RDBMS: Temp space	Random	Database or file system page size	Very high
Multimedia streaming	Somewhat random	64-128 KB	Low
Video streaming/data capture (satellite/laboratory)	Sequential to scattered sequential as threads increase	4 MB 1 thread — 64 KB (many threads)	Depends on capture (high) versus analysis (low)

Host File System Impact

At the host level, file systems also affect application I/O characteristics by determining the minimum and maximum I/O request sizes. File systems intervene between the application and the storage system by buffering and coalescing data.

At the file-system level, the sequential stream of bytes seen by the application is organized into chunks: either blocks or, in some cases, extents. The size of the block depends on the file-system parameters. For instance, on larger partitions, the block size for Microsoft Windows Server 2003 is 64 KB by default.

**Figure 4. File System in the Performance Stack**

File System Buffering

All modern file systems buffer data. Thus, when the application layer requests data that has been buffered, the file system retrieves the data from memory very quickly. Some file systems perform **read-ahead** as well—VERITAS VxFS, and some UNIX-based operating systems such as AIX, for example.

File system buffering is of particular use to file servers, where many users may access the same file. In open systems, file-system buffering is so prevalent that the CLARiiON read cache is rarely hit on a reread.

With an RDBMS, which does its own buffering, file-system buffering adds load to the system that often is of no use at all. Many times, RDBMS systems use raw partitions to avoid the aptly named *double buffering*.

Minimum I/O Size: The File System Request Size

The *request size* is the size of the I/O used to transfer data to disk from memory and vice versa. In most cases, the host OS uses the file-system block size as the request size.

The file-system block size typically sets the smallest possible request size, since the file system rarely performs an I/O smaller than the block size. (The exception is that, in some file systems, metadata describing the file-system structure may be in smaller chunks; however, these represent a minority of the I/O.)

Maximum I/O Size

When designing a system that needs high-bandwidth data access, and in which the application can request large I/O sizes, some work must be done to ensure that large sizes are sent to the storage system.

File systems have a maximum I/O size. This is the largest size with which they will write to physical media. In many systems, the maximum I/O size is configurable, but the tradeoff is that setting a higher I/O size results in increased RAM allocated to system buffers (and thus not available to the application). The typical size is 64 KB to 128 KB, although some OSs—such as Solaris—can do up to 8 MB.

File System Coalescing

File systems can coalesce separate requests that are logically contiguous into one large request. For example, in Solaris, the maximum contiguous (MAXCONTIG) parameter that is used when creating a file system defaults to 1 MB (the value 128 times the block size of 8 KB). Thus, requests can be bundled up until they equal 1 MB, and then dispatched to the storage system. This typically helps bandwidth operations. Coalescing can, however, have unintended (and unhelpful) side effects.

The maximum I/O size works in concert with the maximum contiguous setting.

File System Fragmentation

The blocks in a file system may not be contiguous. Anyone who has defragmented a hard disk is aware of file fragmentation. A file read that is logically sequential might not be a sequential operation for the storage device, but instead a random read of file system blocks.

Fragmentation of files can be corrected with host-based tools. Defragmentation can help the storage system speed up sequential operations, like backups, tremendously: clients have saved *hours* doing backups after defragmenting.

File System Alignment

At times, metadata placed on the logical unit affects performance. File systems that place metadata on the beginning of a disk device cause the user files to be misaligned with regard to the RAID stripe. Consequently, a single application I/O that would have fit evenly on the disks may result in one more **back-end I/O** to the physical drive due to **disk crossings**.

For example, with 16 KB of metadata, a RAID group with a 64 KB stripe depth having a 64 KB I/O written to it would result in two disks written. See Figure 5.

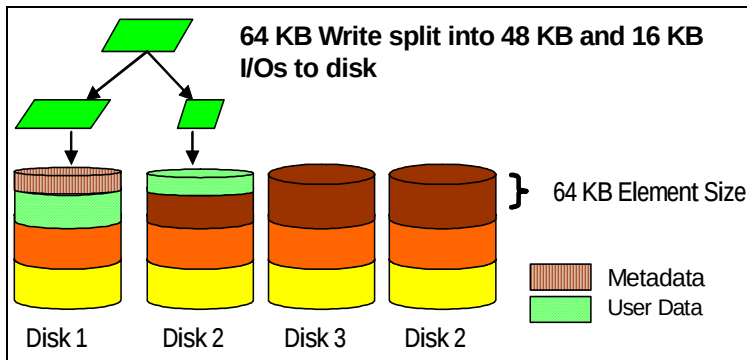


Figure 5. Write Split Across Disks

Figure 5 shows that a 64 KB I/O from the host is split across two disks, thus doubling the latency of an aligned transfer. A 64 KB I/O that follows this one sequentially also requires two I/Os.

In addition, more **stripe crossings** are likely to be incurred. Stripe crossings are more costly with RAID 5, as there is an additional stripe's worth of parity to calculate. File system misalignment affects performance in two ways:

- Misalignment causes disk crossings of small-to-large I/O.
- Misalignment makes it hard to stripe-align uncached, large I/O.

The former issue is more commonly encountered, as very few users attempt to do writes without the intervention of write cache. Most writes hit cache. The caching algorithms will attempt to coalesce and align as much data as possible before dispatch to the drives.

Raw Partitions Versus File Systems

The use of a file system is not mandatory in all cases. An RDBMS can implement tablespaces on raw partitions. This avoids file-system interactions, but in that case, buffering, file management utilities, and backup tools are entirely the responsibility of the application.

Raw partitions have the following advantages over file systems. They:

- Can use larger I/O sizes than most file systems.
- Can use smaller I/O sizes than most file systems.
- Avoid fragmentation issues.
- Are easier to *snap* (that is, perform a SnapView™ function) and remain in coherent state, since file system buffers do not have to be flushed.
- Simplify aligning I/O to the RAID stripe.

Advantages of file systems are:

- More tools are available for managing (moving, backing up, restoring) and analyzing data.
- Backup images are easier to mount. (With raw partitions, the partition must have the exact same **device context** to mount a partition on the backup machine.)
- Applications do not have to be written specifically for them (as opposed to direct access to raw devices).
- Buffering can be advantageous in some cases.

The last point is in reference to very large RAM architectures. For example, in systems with more than 12 GB of system RAM, large data transfers using raw partitions are quick; however, transfers on file systems are nearly as fast and, once buffered, the file system can access the data in its buffers about five times faster than from the storage system. RDBMS servers with large memory sizes (up to 64 GB) may

well perform faster with file systems than with raw devices because of large areas of memory for host caching.

Volume Managers

Volume managers allow system administrators to perform operations on their physical devices at the operating-system level. Volume managers are typically used to provide flexibility that is not inherent in the file system or the storage system. **Logical unit numbers (LUNs)** are the objects of management in the case of Fibre Channel storage.

The operations commonly supported by volume managers fall into several categories:

- Partitioning of LUNs into smaller logical devices
- Aggregation of several LUNs into a larger logical device in order to provide a larger logical device
- Striping of data across several LUNs in order to distribute I/O across many disks

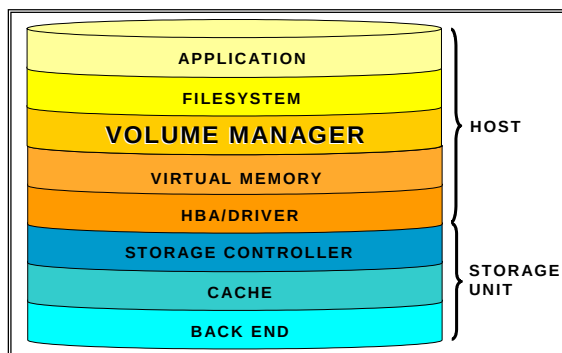


Figure 6. Volume Manager in the Performance Stack

With CLARiiON firmware release 12, all of these operations can now be done on the storage system. The recent release of PowerPath® Version 4.0 includes PowerVolume™, a volume manager. Now the advantages of volume management are available to all PowerPath 4.0 users. How do the two methods of LUN aggregation compare?

There are subtleties in the difference between array-based management and host-based management. For example, concatenating or striping on the host allows the multiple LUNs to be distributed across storage processors. In a small system with few file volumes, this may help balance access to the storage system.

More significant are the effects of striping, which are covered in the following section.

Host-Based Striping (Plaids) and Channel Throughput

A common and outdated technique to storage (which lives on due to debugged infrastructures built around it) is to use a volume manager to build large volumes by striping at the host across modest-sized LUNs (a typical configuration uses 9 GB LUNs). The idea is that multiple Fibre Channels, multiple SPs, and even multiple storage systems can handle I/O to these volumes.

Since the volume is striped across LUNs, and the LUNs themselves are striped across disks, the effect is known as a *plaid*. (A similar effect, with some of the same concerns, can be achieved with metaLUNs. Refer to the *MetaLUNs* section on page 21.)

This approach was critical in the days of SCSI-based storage systems, which were limited to ~20 MB/s per channel. However, a full-Fibre Channel platform, such as the CLARiiON storage system, delivers excellent performance without plaids. In addition, there are tradeoffs and some risks to the plaid approach. Figure 7 shows three different plaid techniques. These are discussed next.

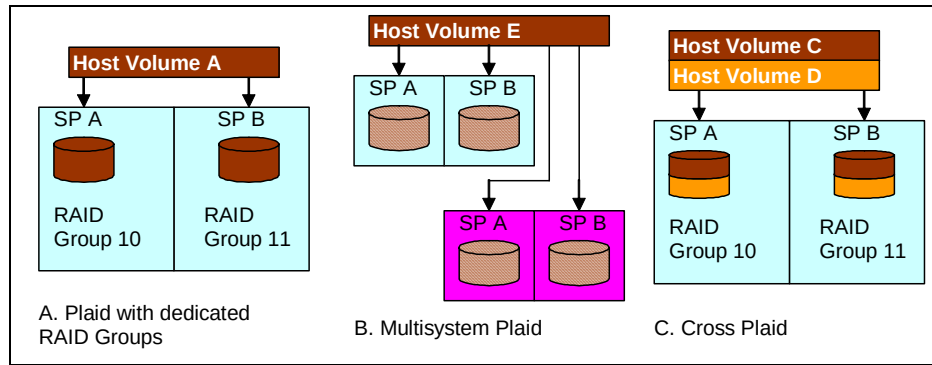


Figure 7. Plaid Types

Plaids with Dedicated RAID Groups

Striping across dedicated RAID groups—groups that the plaid LUNs do not share with other I/O—allows a file system to be distributed across multiple drive groups, storage processors, and processor ports. This is used when I/O sizes are large and access is (typically) sequential. See Figure 7, configuration A.

Unless I/O sizes from the application are very large (4 MB or larger), it is difficult to get better bandwidth from a plaid than from a single dedicated disk group because the key to high performance is operating the drives sequentially.

Accessing the disks in parallel benefits random small-I/O access more than bandwidth. This is due to the concurrency of access. With small, random I/O, there is little dependency between disk operations; each disk acts on one or more I/O, and when that drive is done, all those I/Os are completed to the host. In large I/O requests, the I/O spans multiple drives and does not complete until all drives complete the operation.

Multisystem Plaids

Plaids spanning storage systems (Figure 7, configuration B) are used only when facing *very* high throughput or bandwidth requirements with *very* large file systems. Complexity arises as a fault on one system can affect access rates to all systems.

The Cross Plaid

The primary concern with plaids is that the user usually ends up with RAID groups sharing host volume stripes (a *cross plaid*, see Figure 7, configuration C) causing disk contention and other issues:

- A single fault in a RAID group impacts much more data; all volumes with a stripe in the RAID group are affected. (Performance slows while the group rebuilds.)
- A double fault, as unusual and unlikely as it is, can cause loss of access to multiple volumes in a cross plaid. The risk is small, but the cost of such a failure is great.
- Cross plaids can reduce sequential performance by reducing the I/O size at each LUN and introducing contention for the drives. Both result in fewer sequential operations from the disk drives.

Since drive sizes are growing, and operators demand volume management, cross plaid designs will become more common. A storage system with sufficient cache to manage many LUNs effectively should be used with extensive plaiding.

Host Virtual Memory

“In OLTP environments, performance tuning at the host is—many times—much more important than tuning at the storage system.” — Senior CLARiiON performance engineer

The virtual memory (VM) system is a tool used by the operating system to provide contiguous logical memory spaces for all the applications running on the machine. These logical spaces are managed from a pool of real RAM and from reserved space on disk. When a logical address is accessed, the VM grabs some RAM for it and loads the contents. When an address is not in use, that part of an application’s memory actually resides on disk (paged out). The VM operates on blocks of RAM — much like the file system.

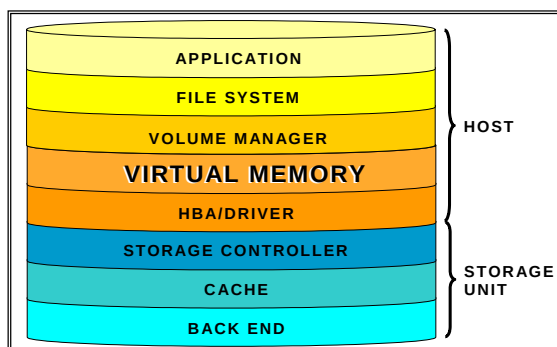


Figure 8. Virtual Memory in the Performance Stack

The virtual memory subsystem on the host can have a profound effect on the efficiency of the applications the host is running. Since the application and host processes encompass so much of the response time, and these processes depend heavily on the VM system, a bottleneck here can cripple the perceived responsiveness of the system.

For example, with IBM AIX systems, the host virtual-memory subsystem *must be tuned* in order to reach the potential of the storage system to deliver data at its optimum.

Table 3 illustrates the effect of a host virtual memory setting tuned for very large memory sizes. This was the result of tuning a host that was running an Oracle database server on the AIX operating system. Bandwidth and throughput are shown as two variables change: storage-system write cache (on/off) and host virtual memory (not tuned/tuned).

The first row shows throughput and bandwidth for storage-system write cache enabled, but with VM not tuned. The second row shows storage-system write cache disabled, but with the host VM tuned.

Table 3. Host File System Tuning Results

Bandwidth MB/s	Throughput IOPS	Storage System Write Cache	Host Virtual Memory Settings
13.5	118	Enabled	Standard
15.2	122	Disabled	Tuned
33.5	195	Enabled	Tuned

Table 3 illustrates that significant application performance can be gained by ensuring that the host memory system is tuned properly. The storage system cannot perform any faster than the host can deliver. In row three, note the synergistic effect of having both VM tuning (increasing the host performance) and write cache enabled (which dramatically enhances the storage-system throughput).

As noted earlier, maximum I/O size is another tuning issue. If the OS is not tuned to allow large I/Os, bandwidth applications are crippled. Watch out when installing RDBMS applications, which often take all available RAM. The reserved file-system buffers may have to be tuned before installing.

Host HBA Effects

Performance and **redundancy** both depend on the number of HBAs installed in a host. However, the primary concern for most hosts is high availability (HA). It takes a lot of CPU power and bus bandwidth to exceed the capability of an HBA. A single 2 Gb Fibre Channel HBA can sustain up to 190 MB/s throughput and about 40,000 IOPS.

If HA is required, there are at least two HBAs and a dual-SP storage system configuration. Because of SP ownership, the LUNs to which the server is connecting can be hosted by different SPs. This gives the host up to 380 MB/s in throughput. See Figure 10.

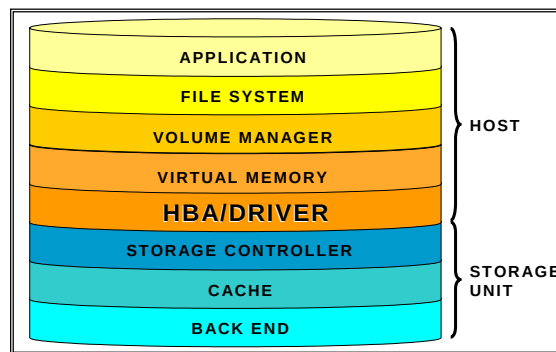


Figure 9. HBA/Driver in the Performance Stack

If a host requires more bandwidth than that to the storage system, HBAs can be added to reach the storage system's maximum bandwidth.

Ensure the storage system can handle the desired bandwidth—check the performance papers published for each model. Also, deploy enough disk drives to deliver the target bandwidth.

In Figure 10, a host with two HBAs uses the SP ownership model to address LUNs from both SPs. In this case, both LUN 0 and LUN 1 are accessed by the same host, but owned by different SPs. Additional LUNs can be addressed through the same two ports (or additional ports) until the bandwidth required is delivered to the host.

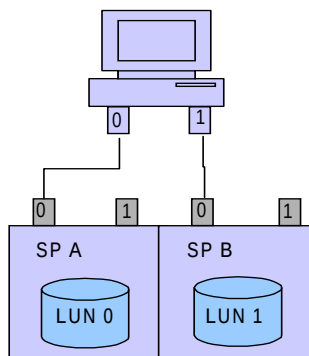


Figure 10. Using SP Ownership HBA Limitations

HBA firmware, drivers, and hardware can all impose limits on the maximum I/O size to the storage system. CLARiiON SPEED members (CSPEED) have access to reports detailing the known limitations. Refer to the Performance community under Avatar for a list of CSPEED specialists.

Failover and Multipathing

A highly available topology is part of the requirement for highly available systems. Another requirement is software that allows the host to connect to all of the storage system's ports—a technique referred to as multipathing. The benefits of multipathing are primarily the ability to fail over from a failed path to an alternate path. **Failover** requires a host component, either host-OS native, such as HP-UX PV Links, or provided by the storage system manufacturer, such as PowerPath.

Another benefit of PowerPath is that given multiple active paths to a LUN, I/O load can be balanced across the storage-system ports and host HBAs in order to avoid bottlenecks.

Refer to the *Considerations for Reliability and Redundancy* section on page 30 for more on PowerPath and redundancy.

Storage Controller Effects

After leaving the HBA on the host, the I/O finally comes to the storage system. The controller's internal bus bandwidth, processor speed, port queues, and buffers all have an effect on performance. The user chooses the appropriate storage system, based on these characteristics. This section covers the factors that impact controller performance. It also addresses differences in the relative performance of the RAID types offered on CLARiiON storage systems.

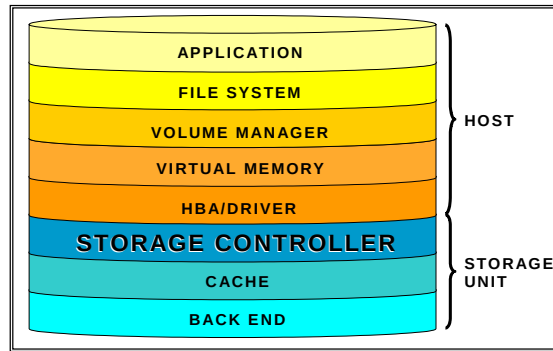


Figure 11. The Storage Controller in the Performance Stack

This section also discusses the user-changeable attributes on the system—chiefly, how the disks are associated and the RAID types used. (A storage controller component—the cache—is important enough to warrant its own section.)

The treatment of RAID by the controller will have an impact. Does it optimize RAID 5 by I/O size and/or sequentiality? When should a particular RAID type be used? What is the impact of applying metaLUNs?

CLARiiON Storage Processors

CLARiiON storage systems are characterized by the storage processors and their packaging.

Storage Processor Enclosure-Based Systems

The CLARiiON storage processor enclosure (SPE)-based systems are built around an enclosure that provides connectivity, cache, and cooling, but no disks. This design offers the highest level of processing power, as all power supply and cooling resources can be dedicated towards servicing a high-powered processing chipset.

SPE-based systems offer the highest level of consolidation, and typically higher, sustained bandwidth capabilities.

Disk Processor Enclosure-Based Systems

CLARiiON storage systems built around a disk processor enclosure (DPE) offer connectivity, cache, control, and up to 15 drives in a 3U package (plus 1U for a supplemental power supply [SPS], if write caching is used). These systems offer the highest density of storage since there is no separate SPE consuming rack space. They typically do not offer as much consolidation as SPE-based systems, however: their controllers share power and cooling with the disks and thus offer less computing power than an SPE-based system of the same generation. Consequently, they do not support as many disk drives as an SPE-based system.

RAID Levels

As this is not a RAID primer, it is assumed that the reader is familiar with these RAID concepts:

- Striping
- Mirroring
- Parity

What is explained is the impact these technologies have on performance.

Parity Versus Mirroring

When writing small I/Os to a parity (RAID 5 or RAID 3) LUN, the controller has to read, calculate, and write a parity segment for every data write operation. This is the RAID 5 *write penalty*. (Refer to *Appendix A: RAID 5* for details.) The total random I/O load on the drives for RAID 5 is:

$$\text{Total Disk IOPS} = \text{Read IOPS} + 4 * \text{Write IOPS}$$

While for RAID 1 or RAID 1/0 it is:

$$\text{Total Disk IOPS} = \text{Read IOPS} + 2 * \text{Write IOPS}$$

Thus, the drives must do more work to absorb a random write on a RAID 5 system. The load on the disk drives will increase faster as host writes increase when parity RAID is used. One would think that writes on a RAID 5 system would be *slower* than a RAID 1/0 system. In fact, this is not necessarily so. If a system has write caching turned on and is not approaching its maximum throughput, the response time for a RAID 5 write is *just as fast* as a RAID 1/0 write.

RAID 1/0 is often suggested for systems experiencing high rates of random write I/O. Why is this?

- The RAID 5 write penalty manifests in slower cache flushing: the cache fills under a lower host load than when using RAID 1/0.
- As the cache fills, the storage system must take action to flush **cache pages**, and this affects response times.

The result is that with RAID 5, the system can sustain a lower overall sustained write load as compared to a system using RAID 1/0. There is another, second-order effect as well: The increased load on the disk drives with RAID 5 can cause contention for reads, thus slowing read response times.

Parity Stripe Optimization

In some performance cases—large and sequential I/O—RAID 5 is a better choice than RAID 1/0. This is because when writing to the same capacity, a parity stripe has fewer drives that need to be accessed. The mirror stripe has to wait longer for all the drives to seek and transfer. This is true, of course, if only the parity RAID stripe can escape the parity write penalty.

The technique CLARiiON uses is straightforward. With write cache enabled, the SP waits until data for an entire stripe is cached before writing it—thus eliminating the RAID 5 write penalty. In this case, the total I/O load on the drives is:

$$\text{Total Disk IOPS} = \text{Read IOPS} + \text{Write IOPS} * (1 + 1/\text{stripe size})$$

While the mirrored RAID set disk load is still:

$$\text{Total Disk IOPS} = \text{Read IOPS} + 2 * \text{Write IOPS}$$

RAID 5 and RAID 1/0 are about equal in read performance. When many processes and threads request data, or asynchronous I/O is used, RAID 1/0 will have an advantage in random reads. Things get more interesting in write performance. RAID 5 outperforms RAID 1/0 in sequential writes of any size, and in stripe-aligned random writes of 256 KB and above. If I/O is not aligned to the RAID stripe, RAID 1/0 is a better choice.

RAID 1/0: The Random Write Leader

In RAID 1/0, the stripe segments are mirrored. A CLARiiON storage system can read from either mirror or primary. Writes are committed to both.

Is RAID 1/0 *faster* than RAID 5? Contrast the mirrored-disk load formula with that for RAID 5. The lower load on the disk drives for random writes allows a system using RAID 1/0 to sustain more writes without saturating the cache. Writes hitting cache are serviced at the same speed.

A secondary effect of RAID 1/0 is that the lower write load on the drives results in less contention for random reads. Since the cache does not buffer random read performance, this effect is noticeable on moderate to busy systems.

Therefore, given equal capacity and a significant random-write load, RAID 1/0 offers performance benefits over RAID 5 when a significant part of the load is random writes. Since RAID 1/0 has almost two times the number of drives available for reads, RAID 1/0 can outrun a RAID 5 implementation if random-read load is very heavy.

RAID 3

RAID 3 offers very high bandwidth reads and writes for workloads that are sequential in nature. RAID 3 also benefits from a large request size. The benefits of RAID 3 over RAID 5 stem from the same effect that hampers RAID 3 in random-access implementations: a dedicated parity drive.

Due to the dedicated parity drive, in sequential reads and writes the disks operate in a highly sequential fashion. In random write operations, the single parity drive is a bottleneck for parity reads and writes.

MetaLUNs

MetaLUNs are LUN objects created from multiple, smaller LUN objects. The intent of a metaLUN is to provide dynamic expansion of a host's capacity without forcing the host to integrate another LUN at the operating system. The effect of using a metaLUN is very nearly that of using a volume manager. MetaLUNs on CLARiiON storage systems can be created from concatenated LUNs or striped LUNs, as can volume manager metaLUNs. MetaLUNs reduce the number of LUN objects to manage on the host and eliminate the need to purchase volume manager software for each host. (MetaLUN capability is a standard feature on CLARiiON CX series storage systems.)

Because a metaLUN can be created from LUNs that share separate disk groups, they are effective in creating storage that is distributed across a very large number of disk drives. The results of this structure are very similar to that of a volume manager plaid.

However, a metaLUN is differentiated from a volume manager volume by the following characteristics:

- MetaLUNs are composed entirely upon a single SP, whereas volume manager volumes can span SPs and storage systems.
- A volume manager creates a separate thread for each stripe element in a plaid, so that single-threaded operations on the host become multithreaded to the storage system (improving bandwidth); this effect is not seen with a striped metaLUN—there is still only a single thread accessing the storage system from the host.
- A metaLUN's configuration is secured internally on the storage system. File system corruption cannot affect a metaLUN's configuration.
- Accessing a metaLUN from another host is simpler than assigning a plaid to another host.
- Creating an array-based snapshot or mirror of a metaLUN is much simpler than snapshotting the components of a plaid.

The performance ramifications of metaLUNs are discussed in detail in *EMC CLARiiON Best Practices for Fibre Channel Devices*. Operational semantics for metaLUNs can be found in *EMC CLARiiON MetaLUNs: Concepts, Operations, and Management*.

The CLARiiON Cache

The most effective single characteristic of a storage controller is its ability to cache. CLARiiON storage systems have a very *smart* cache—much of the CLARiiON competitive advantage is due to its caching algorithms. Maximizing this tool is crucial for top performance.

This section discusses how the CLARiiON cache works, and explains some of its settings.

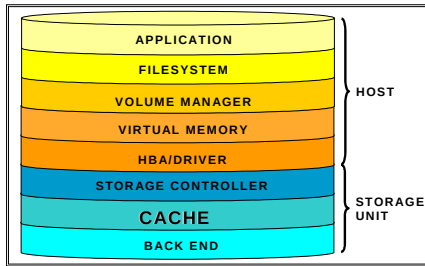


Figure 12. Cache in the Performance Stack

Cache Size and Performance

Any storage system should be a balance of CPU power, cache, front- and back-end ports, and disk drives. Beyond sizing sufficiently to support its supported number of disk drives and ports, the *amount* of cache is not as significant as the system's ability to flush write cache to disk, or its ability to perform prefetch. This section examines each usage case for cache: read cache, read prefetch, and write cache.

Read Cache

CLARiiON storage systems are used in the open systems space, since they do not offer mainframe attach. The typical case in open systems is a file system that buffers read requests. Thus, a CLARiiON system gets very few hits from read cache due to reads being re-requested—those requests are serviced from file-system buffers. Where raw devices are in use, the typical application is an RDBMS, which will do its own read buffering. Reads can be hit from writes that have been cached in the write cache.

As a result, there is little requirement for allocation of read cache for host rereads.

Read Prefetch

Prefetch is the act of reading blocks just beyond those just requested by the host. A CLARiiON system does this when sequential accesses are detected, and places this data in read cache. The host reads a prefetched block only once. As soon as it is read, it can be discarded, because host-level buffering will service rereads. For most users, a modest allocation is sufficient for prefetch—50 MB to 100 MB per storage processor is typical.

Write Cache

The size of the write cache is significant for two reasons:

- Space is needed in which to absorb bursts of writes.
- Data can be kept in the cache to get *hits* from reads and rewrites.

In sustained loads, the most important aspect of write caching is the *flush rate*, not the size of the cache. As long as the system can write out the back as fast as writes coming in the front, the size of the cache is unimportant.

However, bursty writes require reserved cache pages. When a burst of write I/O exceeds the system's ability to flush cache, the storage system needs free cache pages to absorb the incoming data.

The upsides to a larger cache are:

- It absorbs larger bursts of write I/O before it fills.

- The user can raise the lower watermark to get more read and write hits from the write cache.

On each model, the CLARiiON cache is well balanced: with its back-end's ability to flush, its **standby power supply's (SPS)** power capacity, and the **vault** space provided.

Write Cache Mirroring

CLARiiON storage systems mirror the contents of write cache. Unlike some other systems, the mirroring feature cannot be defeated. Running with an unmirrored cache can lead to disastrous effects arising from failure which would otherwise be benign.

For example, in the case of a cable failure between a host and the storage system, if a LUN can no longer be accessed from the active storage processor, the storage system will trespass the LUN to the peer storage processor, which acts as the inactive path for that LUN. This is how every midrange storage system using dual SPs will handle this failure and, in the case of the CLARiiON storage system, the result is uninterrupted service.

However, in a storage system that has had write cache mirroring disabled, the data in the cache on the inactive SP does not match the data in the cache of the active SP. When the trespass occurs, the contents of the LUN will change. If the data which the host had written reflected changes in the file structure, entire files could disappear, and the file system structure would appear invalid. The host would likely lose the ability to address the file system, and would possibly crash.

The Cost of Write Cache Mirroring

Given the risk of data loss with unmirrored cache, why would anyone attempt to run without write cache mirrored? There are several costs associated with all write cache mirroring schemes:

- Increased latency, as the host does not receive an I/O complete message until the mirroring is complete.
- Less cache is available for write cache, as now 50 percent of write cache is used for mirroring the peer's data.
- High-bandwidth applications may exceed the ability of the mirroring channel to transfer data.

CLARiiON addresses the first concern by delivering high-speed architectures built on industry-leading designs. Added to that are many optimizations developed during the years since CLARiiON introduced the mirrored write cache.

The second concern is addressed by offering models with varying write cache sizes, up to the largest write cache in the midrange.

Lastly, LUNs used for high-bandwidth applications can be tuned in order to maximize bandwidth without sacrificing random-access operations.

Activating the Write Cache

Navisphere provides controls for caching. In order to activate write caching, the option must be selected in the storage-system properties dialog box. Caching for a LUN must be activated separately in the LUN properties dialog box (by default, all LUNs have read and write caching turned on). Furthermore, the RAID engine disables the cache if the following conditions are not met:

- An SPS must be present, and it must be fully charged.
- All vault drives must be present; they cannot be faulted or rebuilding (if the Highly Available vault setting is selected—as discussed later).
- Both SPs must be present and functional.
- Both power supplies must be present in the DPE/SPE.
- Both fan packs must be present in the DPE/SPE.
- The DPE/SPE and all **DAEs** must have two nonfaulted **link control cards (LCCs)** each.

In Firmware Release 12 for CX series systems, an option has been added to the storage system properties dialog box **Caching** tab. If **HA Cache Vault** is selected (this is the default), write caching is disabled if a single vault disk fails. If this option is deselected, write caching remains active when a single disk fails. Important: if write caching is active when a single cache vault disk fails, data is at risk if another cache vault disk should fail and power is lost (triple failure).

Refer to the *CLARiiON Mirrored Write Cache Architecture* white paper for details on write cache operation.

Cache Settings

The CLARiiON RAID engine cache is one of the most tunable in the business. Out of the box, it works fine for most applications. But, for applications with atypical performance profiles, or for that extra 5 to 10 percent in performance, the parameters can be tuned. For example, use of caching for any particular LUN is configurable—the read or write cache can be activated on a per-LUN basis.

The following configurable parameters are also available:

- Cache page size (global)
- Allocation of cache—read versus write (global)
- Prefetch—type, amount prefetched, when to disable (per LUN)
- High and low watermarks (global)
- **Write-aside size** (per LUN)

Page Size

When data is stored in the cache, it is stored in chunks called *pages*. The cache page size may be 2, 4, 8, or 16 KB—the same range as most file system and RDBMS page sizes. Ideally, the page size used is as large as possible, but also matches the smallest I/O write size used predominantly by all connected hosts. This helps use the cache space effectively.

Prefetch Settings

The CLARiiON storage system offers fixed and variable prefetch. Fixed prefetch is useful for tightly controlling the amount prefetched and is best when I/O sizes are uniform.

Variable prefetch is the preferred prefetch type for CLARiiON. Variable prefetch uses a multiplier that is applied to the incoming I/O request size to determine the amount to prefetch. Also, prefetch can be done in chunks (called *segments*), controlled by the segment multiplier. The segment multiplier is applied to the I/O request size to determine how large a chunk the RAID driver will dispatch to the disks at one time. If the multipliers are the same, a single request is made. If the segment multiplier is smaller than the prefetch multiplier, several requests are made.

The Limits: Maximum Prefetch and Prefetch Disable

The *Maximum Prefetch* setting is the limit—in blocks—for data requested. The purpose of this limit is to prevent the disks from being busied with prefetch at the expense of other I/O. The default setting is 4096 (2 MB) and can be increased to 4 MB.³ For LUNs with mixed I/O, the defaults are recommended. For LUNs that are under entirely sequential access, the maximums result in higher bandwidth.

The *Prefetch Disable Size* is another limit designed to prevent prefetch operations from monopolizing the drives. I/Os of this size and larger have prefetch disabled. The default is 4097 blocks, so I/O of up to 2 MB causes prefetch to occur.

³ The default for max prefetch was increased from 1024, and the default for prefetch disable was increased from 511, for FC4700 systems with the release of code 08.49.x1.

Example: Variable Prefetch with Prefetch Multiplier = 4 and Segment Multiplier = 2. Maximum Prefetch = 4096 blocks (2 MB)

An incoming I/O of 256 KB will result in a total of $256 \text{ KB} * 4 = 1 \text{ MB}$ total prefetched. Segment size is $256 \text{ KB} * 2 = 512 \text{ KB}$, so two requests of 512 KB are sent to the back end.

Idle Count

The prefetch idle count sets, as a trigger, the maximum number of outstanding I/O for a LUN for which the system will still execute a prefetch. If the outstanding I/O count is greater than the idle count, no prefetch is initiated until the outstanding I/O count drops to below the idle count.

High and Low Watermarks and Flushing

While writes are being executed to LUNs with write cache activated, the cache is being filled, and continually *flushed*. Flushing is the process of writing **dirty pages** from the cache to the disks. The CLARiiON storage system has two global settings called *watermarks*—high and low—that work together to manage flushing.

Flushing can be categorized three ways:

- *Idle flush* — This occurs continually, at a modest rate.
- *High Water flush* — This process is activated when cache utilization hits the high watermark. The storage system dedicates some additional resources to flushing. There is minimal host impact.
- *Forced flush* — A forced flush begins when the cache reaches 100 percent capacity. This significantly impacts host I/O and increases response time.

See Figure 13.

The high and low watermarks work together:

- The high watermark is the level of use at which the storage system starts the high water flush.
- The low watermark is the point at which the storage system stops high water or forced flushing and returns to idle flush behavior. This is the level of dirty pages maintained in order to achieve **cache hits** from the write cache.

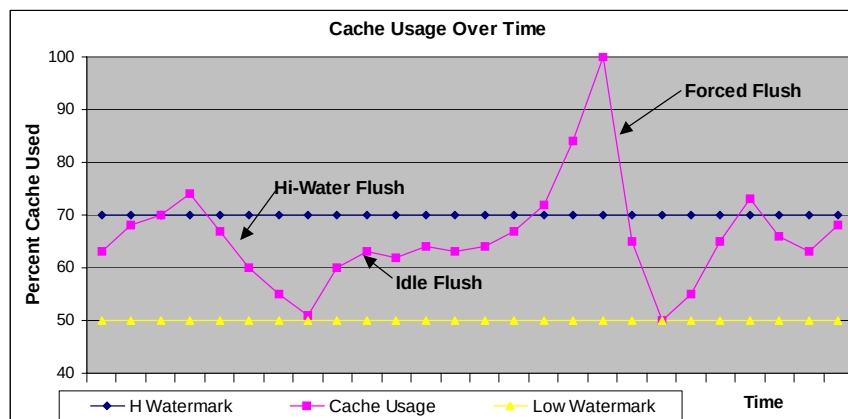


Figure 13. Depiction of Watermarks with Steady-State and Flush Behavior

Write-Aside Size

The CLARiiON storage system has the option to bypass the cache and write data directly to disk. The technique is referred to as **write-aside** in CLARiiON literature. (It is sometimes referred to as *write-through*.) Write-aside is intended to allow large I/Os to bypass the cache. This reduces the impact of large writes on the aggregate write-caching bandwidth.

The sizes configured for write-aside should be large enough that their impact on write cache mirroring may impact other I/O. If the target LUN is a parity RAID type, the I/O must also be large enough to fill the parity stripe so that parity operations do not result.

The LUN parameter *write-aside size* sets the largest I/O that will be write cached. The default is 1023 blocks, so I/O of 512 KB and above bypass cache.

Refer to *CLARiiON Best Practices for Fibre Channel Storage* for more details on using write-aside.

The Back End

The next performance discussion addresses the back end—the disks upon which the data resides, and how they are used.

The storage system's cache can insulate the host from some back-end issues. But when the cache is saturated, or I/O is random, the system's ability to get data to and from the drives is critical.

Configuration of the LUN has some significant effects on performance. A basic tenet of performance tuning is to distribute I/O load evenly across the available spindles. How is this done?

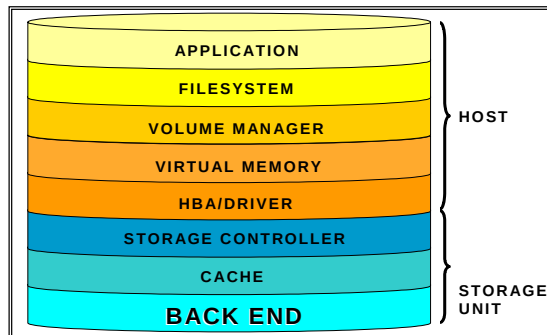


Figure 14. The Back End in the Performance Stack LUN Distribution

On CLARiiON storage systems, sets of disks are associated into RAID groups with Navisphere, the CLARiiON management software. A RAID group contains from two to 16 drives; these drives remain associated until the RAID group is removed. The capacity in the RAID group can be partitioned into 1 to 128⁴ LUNs. LUNs are striped evenly across all the disks of a RAID group. See Figure 15 for a graphical depiction of a RAID group with multiple LUNs.

A RAID group with only one LUN, which takes all the disk space, is called a *dedicated* RAID group. A partitioned RAID group has more than one LUN.

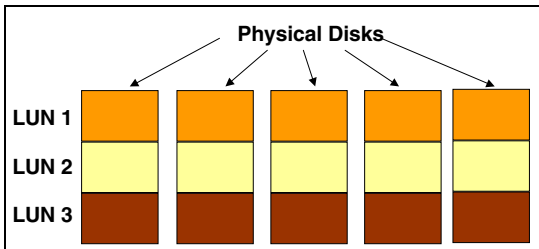


Figure 15. Distribution of LUNs on a Partitioned CLARiiON RAID Group

Note that the LUNs in a partitioned RAID group consist of contiguous sets of RAID stripes. Sequential operations and those using large I/O sizes cause multiple drives to work in parallel, and sequentiality of the drives is maximized. By using sequential stripes, the CLARiiON design avoids the mapping overhead of some drive-level virtualization designs. Refer to the *MetaLUNs* section on page 21 for more on CLARiiON LUN virtualization.

LUN Distribution

The concept of LUN distribution is the basis of I/O distribution. The highest-use LUNs should be distributed evenly throughout the RAID groups. For the FC4700, and for planning purposes only, the

⁴ This was increased from 32 in Release 12.

persistent storage manager (PSM) LUN is considered a high-use LUN⁵. On CX series storage systems, the PSM is hidden and can be ignored.

Take into account the capacity of the LUN when doing distribution planning. Calculate the total GB of high-use storage, and distribute the capacity appropriately among the RAID groups.

Figure 16 shows a simplistic LUN distribution example. Note the LUNs are categorized as high, medium, and low service level, and that they are distributed across the available RAID groups. Note also that RAID group 0 has two high-use LUNs, which are smaller than the single high-use LUNs on groups 1 and 2.

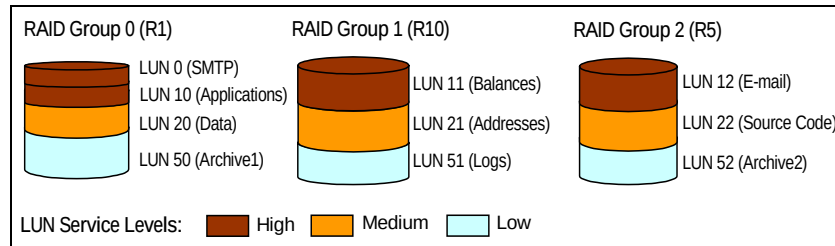


Figure 16. Sample LUN Distribution

Maximum Spindle Counts

A fundamental maxim in performance is that IOPS and bandwidth performance increases as the number of drives in use increases. This holds true for *multithreaded* operations. If the dataset can be distributed over more, smaller drives, performance will be better than when using a few large drives. The key issue here, however, is the assumption that the host has many threads of execution, in order to keep many drives working in parallel.

Minimizing Disk Contention

Disk contention is the term used to describe the condition where different host I/O threads interact at the disk level, causing a high level of seeks relative to the number of I/O involved.

The mixing of file systems expecting sequential I/O with file systems expecting random I/O in the same RAID group results in contention for the disks. The random I/O requests reduce the effectiveness of optimizations for sequential I/O by forcing drive-head movement. Contention for spindles causes seeks, and extracts a cost in increased response time.

Even mixing file systems that do random I/O can cause some contention. Access across a file system is rarely completely random—usually there are active regions that garner the preponderance of activity. Mixing file systems on a LUN causes seeking between the respective active regions of these file systems. However, considering the sizes of modern disk drives, it is inevitable that RAID groups will be partitioned and shared among multiple file systems and data stores. Be aware of the effect contention can have on performance-sensitive activity.

⁵ It is not a high-use LUN, but by planning as such, the user avoids impacts on Navisphere operations.

Stripes and the Stripe Element Size

The stripe element size is the number of blocks the RAID driver writes to a disk before it moves onto the next disk in the RAID group. Multiplying the number of effective drives in the group by the stripe element size results in the stripe size. For example:

- The RAID 5 effective disk count is one less than the number of drives. Five-disk RAID 5 has four effective drives.
- The RAID 1/0 effective disk count is half the number of drives. Eight-disk RAID 1/0 has four effective drives.

The default stripe element size (128 blocks, or 64 KB) is recommended, and should be used except in unusual circumstances. See Table 4 for common RAID-set sizes and their corresponding stripe sizes.

Table 4. Stripe Element and Stripe Sizes

RAID Type and Size	Stripe Size with 64 KB Element
5-disk RAID 5	256 KB
6-disk RAID 5	320 KB
9-disk RAID 5	512 KB
8-disk RAID 1/0	256 KB
16-disk RAID 1/0	512 KB

Parity Stripe Optimizations (Large and Sequential I/Os)

RAID 5 and RAID 3 LUNs can deliver high-bandwidth write performance—in excess of RAID 1/0 speeds—by virtue of stripe-based optimizations. When the RAID engine receives a full stripe, it can generate the parity segment in memory, thus avoiding the parity-read/old data-read disk operations.

Stripe writes are possible when:

- I/O is cached, and the cache allows a stripe to coalesce before attempting to write (typically with sequential writes of any size).
- I/O is aligned with the RAID stripe and large enough to fill the stripe.

Increasing the Spindle Count

An advantage of RAID 5 over RAID 1/0 is that for the same number of disk drives, there will be more effective spindles for data. RAID 5 uses a single disk for redundancy, compared to 50 percent of the disks for redundancy in RAID 1/0.

More spindles for the same load can result in better performance—for example, by dedicating RAID groups to file systems to eliminate spindle contention. This is the inverse of the usual economic argument in favor of RAID 5. (It is cheaper to implement the same amount of storage.)

How Many Disks to Use

There are plateaus in performance where adding disks does not scale workload linearly. At some point, the load on the storage system exceeds the ability of some component to service the demand.

For small I/O in a random-access workload, the system memory resources and CPU speed impose a limit on the number of drives that can be used effectively. In addition, as the number of drives on a Fibre Channel back end increases, an effect known as loop arbitration becomes significant. With more than about 80 drives, the time it takes for a device to arbitrate for control of the loop is a significant part of the transaction time. This applies only to small I/O, since with larger I/O, the disk transfer is always the dominant portion of the service time.

For larger or sequential I/O, another factor in loop behavior becomes important. Due to the very high transfer rates of Fibre Channel disk drives, a fairly modest number of drives can exceed the bandwidth of the SP chipset and the Fibre Channel loop bandwidth.

Disk Types

Performance of the individual disk affects overall system performance. This section addresses the effects of the drive's technology and rotational speed.

Higher rpm drives provide noticeably higher overall random-access throughput and slightly shorter response times than slower drives.

The size of the drive affects the spindle count for a given capacity. In other words, if you have 1 TB of data distributed over 20 disks, and that data is consolidated to 10 disks of the same rotational speed but twice the capacity, the system will have lost the concurrent performance of 10 spindles for servicing that 1 TB. Your hosts may not be able to access data at the same rate they used to.

Fibre Channel Drives

The rotational speed of the drive has a direct impact on random-read performance and a secondary impact on random writes. With random reads, cache hits are rare, and the drive must be accessed. Faster rotational speeds mean less latency with reads.

The rotational speed affects cache flushing for writes, which hit the write cache. Faster drives flush the cache faster than slower drives. The benefit is a higher overall I/O rate supported by the system. If the cache is not being stressed, however, the faster drives do *not* result in faster writes, as the write cache buffers the host from disk operations.

ATA Drives

ATA drives are a relatively new technology in enterprise systems. Early ATA drives were not designed to be used in enterprise-class systems: their focus was low cost and good performance, and they lacked many of the refinements that made Fibre Channel drives the ultimate in performance. ATA drives result in reduced spindle speeds, higher seek times, and an inability to absorb as many operations at one time as the Fibre Channel and enterprise-class SCSI drives.

Initial deployments are intended for relatively low-throughput applications, such as aged data, replication, backup to disk, and staging to tape. ATA drives provide high value, vast amounts of storage, and good bandwidth for these applications. The value per gigabyte of the ATA drives enable data that was once aged to tape to remain online, and encourages the use of more snapshots and clones. The operational effect of these changes is faster recovery of data from a wider range of time frames.

Considerations for Reliability and Redundancy

Not all the design considerations are for performance. Reliability and redundancy must be taken into account when designing a data storage solution.

- Reliability is the ability of the storage system to run for long periods, and during stress, without suffering either hardware or software faults.
- Redundancy is the ability of the system to continue operating through failures without losing host access to data, and data already stored.

The scope for discussing reliability is not as great as performance—or even redundancy. The design decisions on the storage system have been made to ensure a very high level of reliability, so there are few variables on the storage-system side. CLARiiON storage systems' systemic reliability is rated at more than 99.99 percent uptime (exact reliability measurements depend on the model and components used). Data availability can be increased to 99.999 percent by following the redundancy guidelines put forth in this section.

Redundancy is achieved through the intrinsic qualities of the storage system itself and through configurations that take advantage of its features. High-availability attach, for example, is a way to achieve redundancy.

Highly Available Attach

The *EMC CLARiiON Open Systems Configuration Guide* outlines the attach methodologies and equipment that are supported. For each topology in the *Configurations Diagrams* section (in the Benefits table), the guide explains whether it is completely, partially, or non-highly available. It also comments on the cost of each arrangement. See Figure 17, which is excerpted from that document.

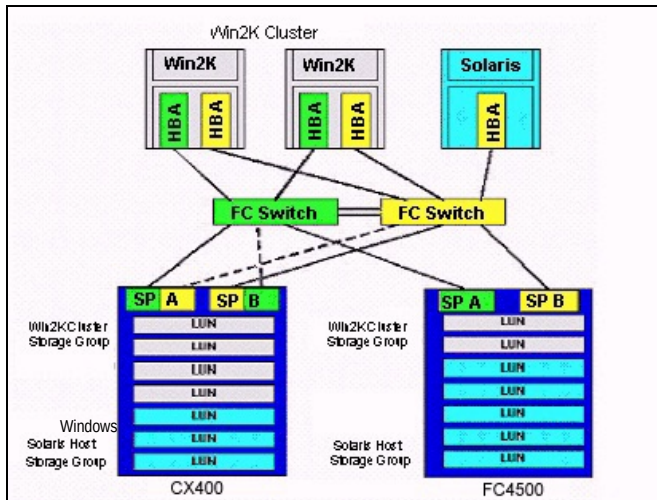


Figure 17. High-Availability Configurations

The two Windows 2000 hosts at the top left are attached in a fully highly available fashion. They have two HBAs each, so the HBA is not a single point of failure. They are also attached to two switches, avoiding another single point of failure. Again, each switch is connected to both storage systems, so the hosts can get to any data from one switch.

The Solaris host, with only one HBA, can survive an SP outage (NDU) on either system because the switches are linked by means of an interswitch link (ISL). It will not be able to access storage in the case of an HBA failure.

If the right-hand switch fails, the Solaris loses access to storage. It is partially highly available. Even in a switched environment, fully highly available attach requires two switches and two HBAs.

RAID Level Considerations

The RAID level determines the amount of redundancy at the disk level. RAID 0 offers no data redundancy at all. RAID 1 and RAID 1/0 offer 100 percent redundancy; that is, up to half of the disks can fail without causing data loss. RAID 3 and RAID 5 offer single-fault tolerance; in this case, one disk can fail and the group will not lose data.

Striping across disks is usually desired for performance and capacity reasons. RAID 0, RAID 3, RAID 5, and RAID 1/0 are all striped RAID implementations. RAID 0's lack of redundancy limits its use. RAID 3 is an implementation-driven solution⁶. Most clients choose RAID 1/0 or RAID 5, so these two choices are examined closely.

RAID 5

On CLARiiON storage systems, RAID 5 is implemented in 3- to 16-disk RAID groups. The main drawback to large groups is the amount of data affected during a rebuild. Rebuild time is also longer with a larger group. Thus, for file systems where slowdowns due to disk failure could be critical, this is a case for fewer spindles per LUN rather than more.

⁶ RAID 3 is typically sensible only for large-block I/O, preferably sequential.

Also, a smaller group provides a higher level of availability since it is less likely that two of five drives will fail, compared to two of nine drives.

RAID 1/0

At the cost of more disks per useable megabyte, RAID 1/0 brings several advantages for redundancy. In comparison to RAID 5, RAID 1/0 does not degrade as significantly during a rebuild after a single fault. Compared to RAID 5:

- A RAID 1/0 LUN of equal capacity rebuilds faster.
- A RAID 1/0 LUN suffers less performance degradation during rebuilds.
- A RAID 1/0 of $n + n$ disks survives n disk failures—provided they are all secondaries (or primaries).

Binding Disk Groups across Disk Shelves

A disk-array enclosure (DAE) in a CLARiiON storage system will be attached to a pair of Fibre Channel loops, one to each SP. Other DAEs will daisy-chain from the first. Depending on the CLARiiON model, you will have one or more such redundant buses in the storage system. Generally, except for some cases of very large disk groups, there is no particular advantage to binding parity groups across multiple DAEs.

For ramifications of binding across back-end buses, refer to *EMC CLARiiON Best Practices for Fibre Channel Devices*.

Summary

Performance—defined by throughput, bandwidth, and response time—is affected by various layers, which interact in order to provide data storage to a user's application. From the OS and host (where the application runs and the file system interacts with volume managers) to the storage-system cache, actions on the data stream cause coalescing, breaking up on I/O, and recoalescing—all of which affect how efficiently the storage system can operate. Being aware of these activities will help with the preparation of your storage network.

Redundancy—the ability of a storage network to deliver service despite a failure—is accomplished through storage-system design, data network design, the use of software like PowerPath, and by choosing the RAID level appropriate for your data's service level.

Glossary

These terms are specific to storage-system technology.

back-end I/O — Operations between the RAID engine and the drives.

bandwidth — A measure of storage-system performance, as measured in megabytes per second (MB/s). It is of most interest for applications that perform I/O in large requests (larger than 64 KB) at relatively low IOPS.

buffering — When an application or file system retains data for reuse, it is referred to as buffering. The technique is very similar in effect to storage-system caching. However, where the storage system uses a specific hardware cache, a file system or application manages buffers in allocated host memory.

burst/burstiness — The characteristic of I/O, which over time is highly variable, or regularly variable, with well-defined peaks.

cache hit — A cache hit occurs when data written from or requested by the host is in the cache (due to prefetch or previous write to cache). This is a very fast way to service I/O.

cache page size — The unit of cache allocation: 2, 4, 8, or 16 KB. Ideally, it matches the smallest average I/O size seen by the storage system.

coalescing — The bundling of smaller I/Os at the file-system level into larger ones. This is accomplished if there is some data locality (for example, the blocks are close together in the file system).

crossing (disk) — If a back-end I/O is not contained in a single stripe element, a *disk crossing* is incurred, which degrades performance. Note that the typical stripe element size (128 blocks = 64 KB) is smaller than most I/O, so a crossing is more likely from a misaligned file system than a large I/O.

crossing (stripe) — If a back-end I/O is not contained in an entire stripe, a *stripe crossing* is incurred, which takes more time (slower performance). See also **crossing (disk)**.

decision support system (DSS) — Decision support systems are built on top of databases, often with historical data. The DSS finds trends, predicts future behavior, and so on. Typically, the DSS application performs reads only, taking in large amounts of data with large I/O sizes. Access tends to be sequential. See also **online transaction-processing system (OLTP)**.

device context — A file system builds its images of storage from devices such as disk and tape devices. The device context is the name the operating system builds for such a device. The standard context with UNIX operating systems is a pattern such as `/dev/dsk/c1t0d2` where *c*=controller, *t*=target and *d*=LUN.

disk-array enclosure (DAE) — The chassis element that holds the Fibre Channel drives. Fifteen drives fit in a DAE (10 in FC series systems). It has dual-shared power supplies and an integral fan pack. In CX series systems, the first DAE is always powered by a supplemental power supply.

disk processor enclosure (DPE) — The chassis element containing the FC4700/CX400/CX200 SPs, vault drives, power supplies, and fans. Host and disk Fibre Channel ports are located on the DPE. It has dual-shared power supplies and is always powered by a supplemental power supply (SPS). See also **supplemental power supply**.

dirty page — A page of write cache containing data written by the host but not yet flushed to disk.

failover — The process by which access to a LUN via a failed path is changed to a functioning path. This requires more than one physical connection between the host and storage system.

flush — The process of writing data held in the write cache to the disks.

latency — The time required for a disk drive to rotate the desired sector under the read head.

link control card (LCC) — The field replaceable unit (FRU), which links DAEs using a Fibre Channel link.

logical block address (LBA) — The address on a physical device where a block of data is located. CLARiiON systems track LBAs for a LUN in order to detect sequential access patterns.

logical unit number (LUN) — Hosts access storage using logical unit numbers, which are exported by a SCSI target. The actual storage object is a logical unit (LU), but invariably LUs are referred to as LUNs, even though this is incorrect.

metadata — Any data used by the partitioning process or file system during housekeeping. The information stored by applications is referred to as user data.

metaLUN — A LUN object built by striping or concatenating multiple LUN objects. Introduced in CLARiiON storage systems with firmware release 12.

MR3 write — Action the CLARiiON RAID engine performs when an entire RAID 5 stripe is collected in the cache and written at one time. Refer to [Appendix A: RAID 5](#).

online transaction-processing system (OLTP) — Multiuser systems, backed by a database, that handle many small transactions. OLTP systems are characterized by a high load of transactions per second, and transactions consist of small (typically <2 KB), random changes to two or more tables. See also **decision support system (DSS)**.

prefetch — A caching method by which some number of blocks beyond the current read are read and cached in the expectation that a read in the near future will request those blocks. This is the way CLARiiON read cache works; it benefits only sequential I/O in any significant fashion.

RAID group — An association of two to 16 drives. All drives in a RAID group share the same RAID type. LUNs are partitioned from the RAID group and span all disks in the RAID group.

random — I/O that is random is written to locations that are distributed across the file system or partition. The location of one I/O is not related to the location of the previous or following I/O. See also **sequential**.

read-ahead — See **prefetch**.

redundancy — The ability of the storage system to continue servicing data access after a failure.

reliability — The ability of the storage system to run for long periods, and during stress, without suffering either hardware or software faults. Reliability is measured in mean time between failures (MTBF), or in uptime percentage. The availability for an FC4700 is between 0.9999 and 0.999942, which translates into expected downtime of between 30 to 53 minutes per year.

request size — In a file system, the size of the block actually read from the disk; this often differs from the application's request. The file system concatenates the application's request from the blocks it reads from the physical media.

response time — The cumulative time for an I/O completion as measured from the host.

saturation — The condition in which a storage system is loaded to the point that adding more I/O dramatically increases the response time for any arbitrary request. It usually marks a bottleneck in the design or the limits of a balanced design.

sequential — I/O made of separate packets that are being written or read, one immediately after the other, in the file system or partition.

spike — A sudden, sharp, and significant increase in load on the storage system. It can be caused by temporal factors, such as a busy period, or by administrator action, such as a database checkpoint operation.

stripe/strip element — RAID algorithms achieve speed and reliability by distributing sequential chunks of storage across many disks. The number of blocks written before I/O jumps to the next disk in the group is the *stripe element* size. The number of disks in group*stripe element size = size of the RAID *stripe*. The stripe size is more commonly used in various calculations, but the stripe element size is crucial as well.

storage processor enclosure (SPE) — The chassis element containing the CX600 SPs, power supplies, and fans. Host and disk Fibre Channel ports are located on the SPE. It has dual-shared power supplies and is always powered by a supplemental power supply.

supplemental power supply (SPS) — A battery that is used to hold up the SPE and first DAE or DPE in a power-fail situation. This allows the cache to be saved to the vault.

throughput — The performance term for measuring I/Os over time; usually measured as I/Os per second (IOPS).

vault — Special area on disks 0 to 4 of the first DAE (CX600) or DPE (CX400, CX200), and disks 0 to 9 of the DPE in FC series systems. This is where the contents of the write cache are stored in a power fail.

write-aside — Bypass of the write cache, where the RAID engine dispatches a write immediately to the disks.

write-aside size — The largest request size, in blocks, that will be write cached for a particular LUN. It can only be changed with the Navisphere CLI `chg1un` command.

Appendix A: RAID 5

RAID 5 (like RAID 3) is well known for providing data protection with economy. By removing the need to mirror data (which doubles the cost of disks in a storage system), parity-based RAID techniques brought data protection to many more accounts than was possible with mirroring.

The Parity Write Penalty

The requirement to compute parity for each stripe of data laid down on the disks is the cost of parity-based RAID schemes. When I/O is small, and only one stripe element must be written out, this parity calculation represents significant overhead. Figure 18 shows the steps the storage system takes to execute an I/O to a single disk in a parity RAID scheme.

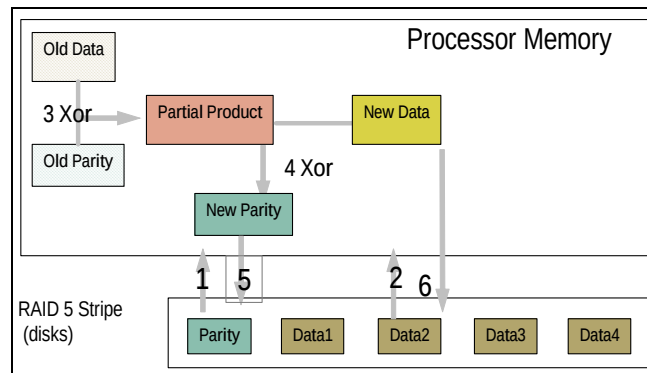


Figure 18. RAID 5 Write Penalty

The steps are:

1. The stripe element holding the old parity is read into memory.
2. The old data (which will be replaced by the incoming data) is read off its disk into memory.
3. In memory, the old data and old parity are XORed to create a partial product.
4. The new data and the partial product are XORed to create the new parity for the stripe.
5. The new parity is written to the disk. (A failure at this point allows the new data to be calculated based on the CLARiiON patented on-disk time stamp.)
6. The new data is written out.

These six steps represent a total of four disk I/Os—two more than would be necessary in a mirroring scheme. This increases latency and introduces the RAID 5 write penalty. It also applies to RAID 3 for small I/O.

CLARiiON RAID 5 Optimizations

In a parity-based RAID scheme, if the entire stripe is sent from the host, the parity must be calculated. However, no other processing must be done except writing the stripe to disk. See Figure 19.

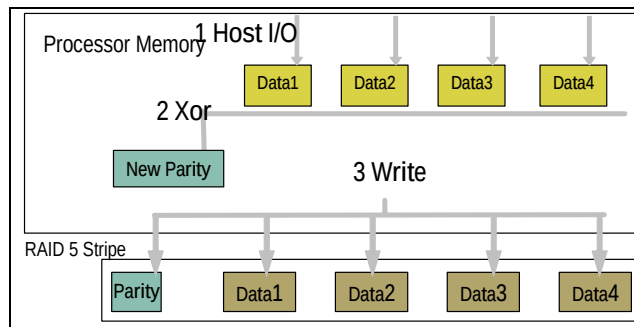


Figure 19. RAID 5 MR3 Write

The steps are:

1. Stripe is received from the host as a large I/O or many sequential I/Os.
2. Entire stripe is XORed in memory to produce a new parity segment.
3. Entire stripe, including new parity, is written to the disks.

In comparison to a mirrored stripe scheme, when writing a full stripe, the parity-based RAID implementation writes to $N+1$ disks. The mirrored scheme writes to $2*N$ disks. Thus, the parity-based RAID approach has less latency—waiting for fewer disks—if the entire stripe can be written.

The CLARiiON RAID 5 optimizations are based around this effect. The optimizations work to delay writing data to the disks until a RAID 5 stripe has filled, at which time a modified RAID 3 write (**MR3**) is performed. The CLARiiON storage system does this by coalescing data in the write cache two ways:

- Large I/O stripe detection: If a large I/O is received, the RAID engine detects whether it can fill a stripe, and writes the stripe out in MR3 fashion.
- Sequentiality detection: The RAID engine detects data being written to the LUN that is sequential and delays flushing cache pages for a stripe until the sequential writes have filled the stripe.

In either case, partially filled stripes will result in additional disk load: either parity operations occur, or the stripe is filled by reading the stripe off the drives (backfilling) in order to complete the stripe in memory. At that time, the parity is calculated and the entire stripe written out.

Because the parity stripe involves fewer disks than a mirror stripe, for large I/O writes (256 KB and above) and in sequential writes, a CLARiiON RAID 5 LUN can provide faster performance than a RAID 1/0 LUN.