White Paper

EMC CLARiiON Fibre Channel Storage Fundamentals

Technology Concepts and Business Considerations

Abstract

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

September 2007

Part Number H1049.2

Table of Contents

Executive summary 5
Introduction 5
Audience 5
Terminology 5
Overview of disk storage and RAID 5
Understanding disks 5
Understanding RAID 6
Considerations for performance 7
Visualizing the performance problem 7
Defining performance 7
Characterizing I/O 8
Application design 10
Why tuning host software is important 11
Application and I/O characterization 11
Host file system impact 12
File system buffering 13
Minimum I/O size: The file system request size 13
Maximum I/O size 13
File system coalescing 13
File system fragmentation 14
File system alignment 14
Raw partitions versus file systems 14
Volume managers 15
Host-based striping (plaids) and port throughput 15
Host virtual memory 16
Host HBA effects 18
HBA settings 19
Failover and multipathing 19
Storage controller effects 20
CLARiiON storage processors 20
RAID levels 20
MetaLUNs 22
The CLARiiON cache 23
Cache structure 23
Cache and performance 23
Write cache mirroring 24
Activating the write cache 24
Cache settings 25
The back end 27
LUN distribution 28
Keeping drives busy 28
Minimizing disk contention 29
Stripes and the stripe element size 29
How many disks to use 29
Disk types 30
Considerations for reliability and redundancy 30
Highly available attach 30

RAID level considerations 31
RAID 5 32
RAID 1/0 32
RAID 6 32
Using the cache 32
Conclusion 33
Glossary 33
Appendix: RAID 5 36
The parity write penalty 36
The parity write penalty and RAID 6 37
CLARiiON RAID 5 optimizations 37

Executive summary

This white paper provides the building blocks for understanding EMC's more advanced performance-oriented white papers and courses.

We have found that understanding basic performance concepts is very helpful to the end user, the EMC account team, and the technical consultants involved in design and implementation.

This knowledge helps you anticipate what your needs are and avoid problems. Correction of a design is far less expensive in the design stage than after implementation. Thus, education upfront will pay off in measurable terms.

Introduction

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

Although this paper focuses on Fibre Channel systems, most of the concepts presented also apply to iSCSI systems. The differences between these two protocols are covered in more advanced documentation.

Audience

The intended audience of this paper is broad. Anyone involved in the procurement, design, or implementation of a block-storage system can benefit from this paper.

Terminology

The first time performance concept terms are used they are *italicized.* These words are defined in the "Glossary" section.

Overview of disk storage and RAID

Mechanical disk storage has a long history. Performance has traditionally been focused on seek time and burst transfer rates. However, disks used in arrays, especially in centralized storage with multiple array sets, add subtleties to the analysis. Burst rates are not typical in a bus or switched arrangement and the bus itself will limit bandwidth. Latency is added by the controller in some cases and drastically reduced (via cache) in others.

Understanding all these relationships and how they interact is the subject of more advanced material. But the basics of disk and RAID performance, and their impact on system performance, can be summarized with a few straightforward concepts.

Understanding disks

There are many subtleties to disk performance, but the following discussion introduces you to the key concepts.

A hard disk drive can execute only one operation at a time. If more than one request for an operation is sent to a disk, they are written to a queue where they are stored until disk is able to execute each request. The core of all performance is the disk's *service time*, which is the time it takes the disk to execute a request once it has received the request. Queues may be stored on the disk, or, if the disk does not offer queuing, they are stored on the controller. The amount of time that elapses from the time a request is written to the queue until the disk executes the request is the average *response time*, and is represented by the following equation:

```
Response time = (queue length + 1) * service Time
```

This concept is illustrated in Figure 1:

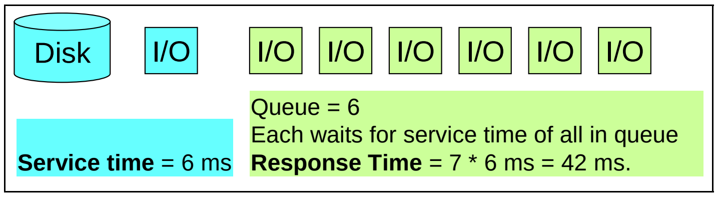

Disk
I/O
I/O I/O I/O I/O I/O I/O
Queue = 6
Each waits for service time of all in queue
Response Time = 7 * 6 ms = 42 ms.
Service time = 6 ms

Figure 1. Disk service time and response time

The classic analogy is that of a busy cash register line in a retail store. The time it takes you to exit the register line is the sum of all service times of those ahead of you in line (including the person at the register, hence the +1). Obviously, if lines are long, it takes longer to get out of the store. In this case, the manager might open more registers, which reduces the queue length. The result is that everyone gets out of the store faster. The solution is the same for an overloaded disk array: Add more disks.

When a disk reaches a queue size between 6 (for slower drives) and 20 (for faster drives), it is approaching its performance limit. Some optimizations occur as the queue lengthens (which softens the saturation effect by lowering service time), but overall, the disk queue depth becomes the dominant factor in host response time. Only a very deep cache (as on EMC Symmetrix®) can keep host response time acceptable as queues pass 20 to 24.

The controller performs certain operations that can make the disk's job easier. Any time the disk spends seeking or waiting for a sector to spin under its head is wasted time because no data is being transferred. When data on successive sectors of a disk are accessed, no time is wasted seeking. The access pattern of reading or writing successive sectors on disk is called *sequential access* and the service time for sequential access is very low**.** The service time for a sequential read can be under 1 ms, so a much higher queue can be tolerated than for a random read (which has a service time of 4 to 6 ms). Many optimizations try to maximize the sequential nature of disk drives.

Understanding RAID

Modern disk storage systems use RAID techniques to provide higher performance (by using multiple disks in parallel) and higher redundancy (the ability to survive a drive failure with data intact). In some situations, there are tradeoffs between performance and redundancy.

The basic components of RAID are described in the following list:

- Striping is a method of spreading data across multiple drives in order to use the drives in parallel; this allows more data to be processed in a shorter amount of time. Striping with no redundancy (data protection) is called RAID 0.
- Mirroring is a technique by which two copies of data are stored on different disks. Thus, in the event of a disk failure (or media failure), the data is intact on the surviving disk. (RAID can also mirror entire stripes of disks to stripes on other disks.) Single-disk mirroring is referred to as RAID 1.
- Parity is a method of protecting a stripe from disk or media failure without paying the storage cost of an entire mirrored stripe set. For example, in RAID 3, a single drive is added to the stripe to hold parity—a mathematical construct that allows re-creation of a missing segment of data.

All other RAID types are derived from these basic concepts. For example, RAID 1/0 is a mirror of a stripe. In RAID 5, the parity data rotates among the drives in the set, so that parity operations do not rely on a single drive. RAID 6 is a parity protection scheme that uses two disks per stripe for parity data.

These concepts deserve some study. A good knowledge of RAID helps in understanding CLARiiON's implementation of RAID. It is important to note that *every RAID implementation is different and the* *designer cannot make assumptions about how a particular system will work*. The purpose of this paper and the more advanced *EMC CLARiiON Best Practices for Fibre Channel Storage* white paper (on EMC Powerlink®) is to explain implementation-specific factors. For example, details on the CLARiiON RAID 5 optimizations are included in "Appendix: RAID 5."

The RAID types commonly used with CLARiiON systems are RAID 1 (mirrored disk), RAID 1/0 (striped mirror), RAID 5 (rotating parity), and RAID 6 (dual parity). RAID 3 is used primarily for high-bandwidth applications. RAID 0 is rarely used as it offers no protection from disk failure. Note the RAID "number" has no relation to the number of disks in a RAID group. For example, RAID 5 groups on CLARiiON can contain three to 16 drives.

Considerations for performance

This section focuses on the throughput and responsiveness of I/O subsystems, which is EMC's primary concern. How important is I/O 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 help explain this, a layer diagram (Figure 2) is used throughout this 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 to be nonintuitive, but the storage system cannot be optimized until the application and host behavior have been characterized.

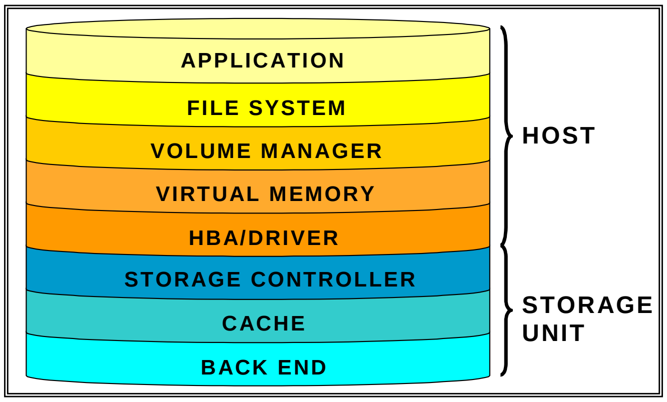


Figure 2. Performance stack

Defining performance

You can measure performance in several ways. The most common terms used to measure performance are explained below. They are related but different perspectives on measuring I/O performance.

- *Throughput* — The number of individual I/Os the storage system can process over time. It is measured in I/Os per second (IOPS). Using the grocery store analogy, this is like looking at the number of *individual items* a cashier checks in an hour.

- *Bandwidth* — The amount of data the storage system can process over time. It is measured in megabytes per second (MB/s). This is analogous to the number of *cartloads* a cashier checks in an hour.
- *Response time* — The interval of time between submitting a request and receiving a response.

Of these terms, response time is the measurement most visible to users. All layers of the performance stack affect response time, so it is an important measure. More importantly, however, *response time is what users complain about*. Slow screen refreshes and batch files that take too long are common complaints.

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 increases, throughput and bandwidth increase. However, when a system approaches the limits at which it can operate, response times increase exponentially. Figure 3 shows a chart from an *online transaction-processing (OLTP)* performance profile. Response time is plotted as the load increases for a particular configuration.

Note that, for any given load, increasing the number of disk drives reduces the response time – a direct result of the principle illustrated in Figure 1. Performance improves as we add drives, which is known as *scaling.*

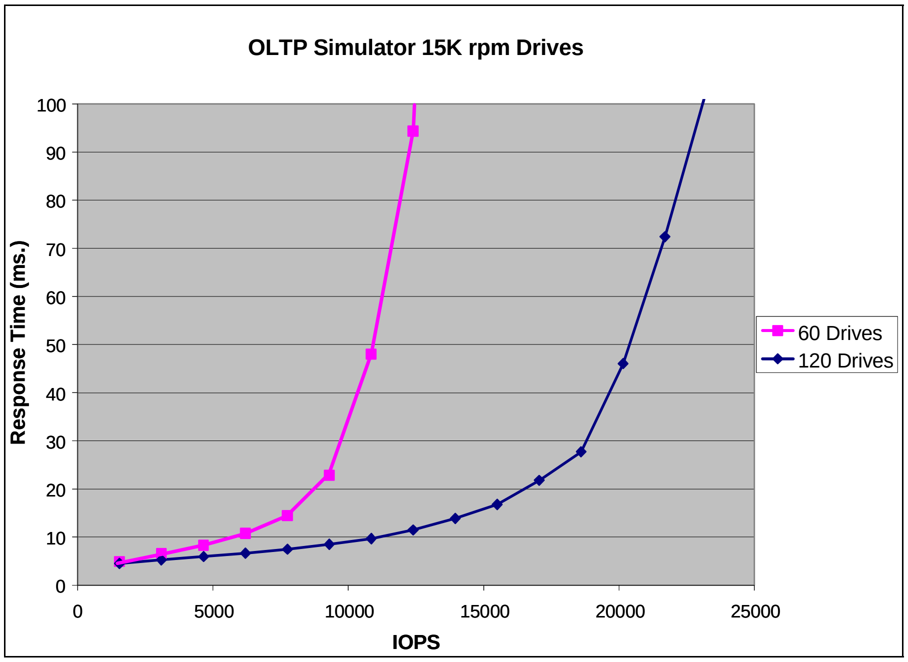


Figure 3. OLTP simulation

Note that the plot in Figure 3 reaches a point where its response time increases dramatically. This is the *saturation*, a point where the queue depth of the drives is so high that adding load simply creates higher response times. Where the 60-drive line saturates, the 120-drive line is still increasing gradually, until a higher I/O rate saturates the larger number of drives.

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 that are spread across the addressable capacity of the LUN. Sequential refers to successive reads or writes that are physically contiguous—one *logical block address (LBA)* after the other.

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.

If you know your access pattern is random, you will have a better expectation of how the storage system can perform, and that the chief dependency in your configuration for good performance is the number of disk drives.

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 (controller cannot accurately predict for prefetch); requires multiple fast disks for good performance.	Caching is effective, resulting in response time better than disk response times (until cache is full).
Sequential	Caching is extremely effective because the controller can correctly predict accesses and prefetch data into cache; 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 low penalty.[1]

I/O request size

Finally, the *request size* of the I/O is important. Larger I/Os take longer to transmit over 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 I/Os. The response time of each large transfer is longer than the response times for a single smaller transfer, but the combined service times of many smaller transactions is 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. In Figure 4 the inverse relationship of I/O size to throughput is charted with the positive relationship of I/O size and bandwidth.

[1] Refer to "Appendix: RAID 5" for details on RAID 5 optimizations.

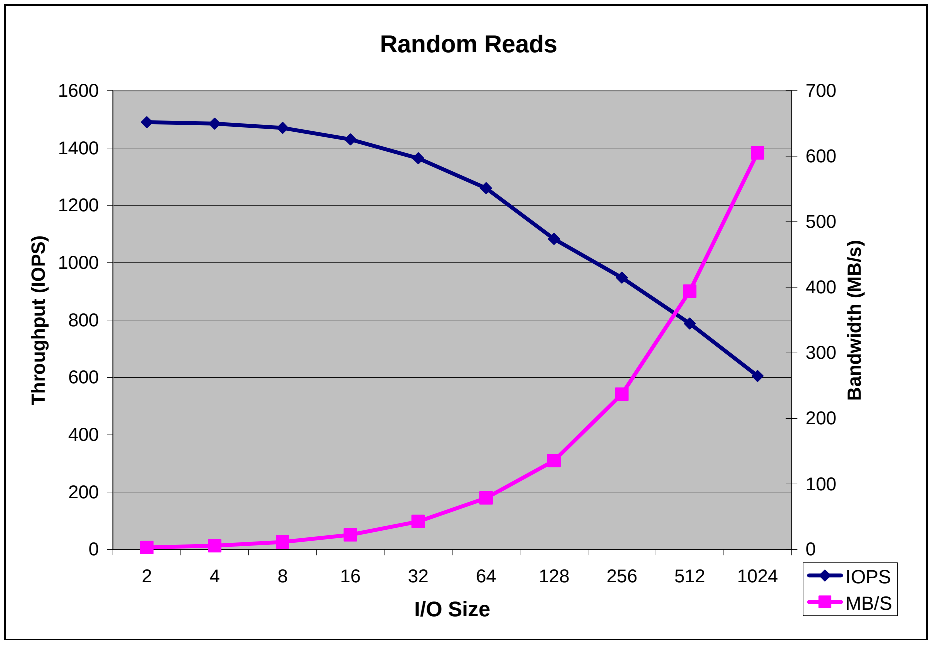


Figure 4. I/O size, IOPS and bandwidth (MB/s)

Random I/O is typically dependent on disk drive performance, for reasons noted above. However, as I/O size reaches 16 KB and above, high rates of I/O may begin to reach the limits of the ports and the memory system on the controller.

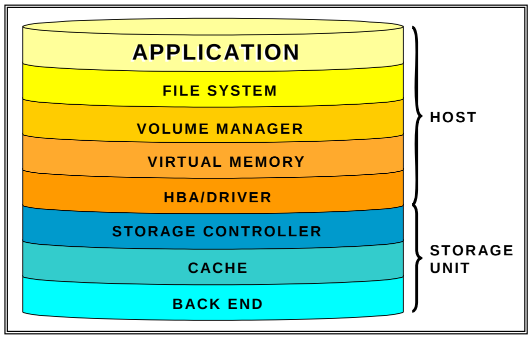
Bursts and busy periods

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

Typical causes of bursty behavior are database checkpoints, busy times of the day, and file system flushes.

Application design

This section describes the application layer of the performance stack—the end user of the storage. The application introduces the need for storage. Without an application, there would be no data to store. 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

Figure 5. Application layer

Why tuning host software is important

All of the previously mentioned application characteristics affect response time. In the performance stack (Figure 5), 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.

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 the 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, 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, creating temp files for data warehouses, and backing up files. Transaction log activity is sequential.

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

Reads versus writes

Typical read/write ratios are:

- **Online transaction processing (OLTP)** — 67 percent reads and 33 percent writes. Few table scans.
- **Email** — Very similar to OLTP. Microsoft Exchange Server 2003 is noted for its very random 60/40 mix of reads and writes. Exchange Server 2007 is closer to a 50/50 mix.
- **Decision support system (DSS)** — Also known as data warehouse or business intelligence. I/O load is 80 percent 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 are 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 and the use of ASM soft striping.

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 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, or application response times will suffer during busy periods. The disk group hosting a bursty file system must also have performance held in reserve. Drives hosting bursty data should be no more than 70 percent utilized under nonpeak conditions.

Application buffering and coalescing

Some applications, such as a relational database management system (RDBMS), buffer data using host memory. Buffering is the process of putting data about to be written into a special area of 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 which data—or which type of data—is best to retain in its buffers. Therefore, buffering benefits performance. Application buffering is different than file system buffering or storage system caching, neither of which makes use of application design to decide which data to retain in the buffers or cache. Often, applications are adapted to bypass file system buffering.

A host file system and some applications also *coalesce* writes. Coalescing usually helps performance by reducing the transfers to the storage system, though the larger I/O has a longer response time than any one smaller request.

Applications summary

This section examines typical applications and their characteristics. A moderate or high percentage of writes implies that an application requires some consideration for write throughput design. That is covered in more detail in *EMC CLARiiON Best Practices for Fibre Channel Storage*.

Table 2. Typical applications and their characteristics

Application processing	Seek type	I/O request sizes	Percentage of I/O as writes
Microsoft Exchange	Very random	4 KB	Moderate-high
Microsoft Exchange Backup	Random	64 KB	Very low (source)
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	64 KB to 1 MB (Oracle ASM)	Low
RDBMS: Online (transaction) logs	Sequential	512-byte+	High, except for archiving process
RDBMS: Temp space	Random	Database or file system page size	Very high
Multimedia streaming	Somewhat random	64 KB to 128 KB	Low
Video streaming/data capture (satellite/laboratory)	Sequential to scattered sequential	64 KB+. Very application-dependent, resolution-dependent as well.	High

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.

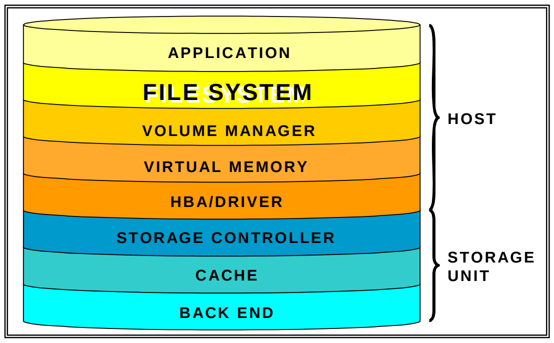


Figure 6. File system

At the file system level, the sequential stream of bytes seen by the application is organized into chunks called blocks, clusters, or allocation units. The size of the block depends on the file system parameters. For instance, on larger partitions, the block size for NTFS (used in Windows systems) varies from 512 bytes to 4 KB and depends on the size of the partition being formatted.

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—such as VERITAS VxFS and some UNIX-based operating systems such as AIX.

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, database 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 operating system (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 they will write to physical media. In many systems, the maximum I/O size is configurable; 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 most file systems can do up to 1 MB or larger (at which point other factors may intervene).

File system coalescing

File systems can coalesce separate requests that are logically contiguous into one large request. For example, in Solaris, the maximum physical I/O size (`MAXPHYS`) parameter works with another parameter, `MAXCONTIG,` that is used when creating a file system. With both set for 1 MB allocations (128 blocks of 8 KB), sequential requests for large files can reach 1 MB, and then be dispatched to the storage system in a large I/O. This typically helps bandwidth operations.

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 detection is built into many operating system utilities; without these utilities it is hard to detect fragmentation without array-based analytical tools.

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. Volume managers that place metadata on the beginning of a disk device cause the file system that follows the metadata to be misaligned with regard to the RAID stripe. Consequently, some application I/O that would have fit evenly on the disks may result in additional *back-end I/O* to the physical drives 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 writes to two disks, as shown in Figure 7.

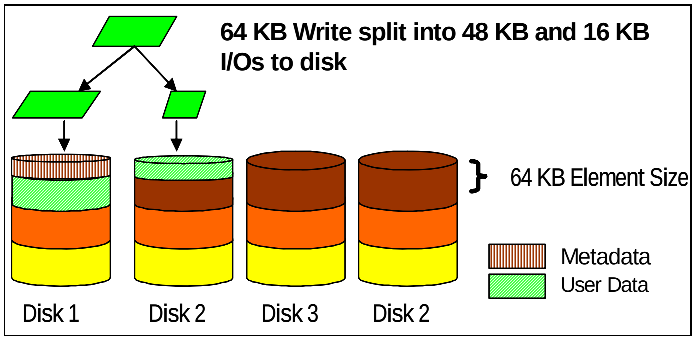


Figure 7. Write split across disks

Figure 7 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, the application is entirely responsible for buffering, file management utilities, and backup tools.

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 snapshot (that is, perform a SnapView™ function) and remain in a coherent state, since file system buffers do not have to be flushed.
- Simplify aligning I/O to the RAID stripe.

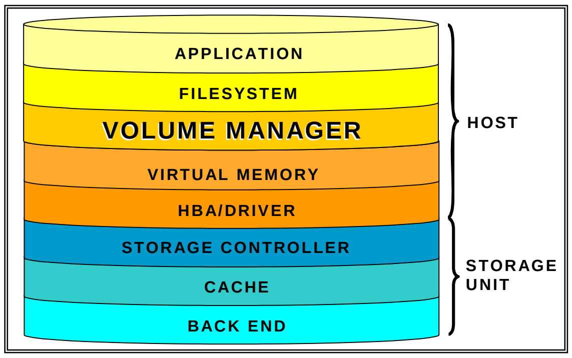
Advantages of file systems are:

- 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.)
- Massive buffering can be advantageous in some cases.

The last point is in reference to very large RAM architectures. RDBMS servers with large memory sizes (up to 64 GB) may 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 units* (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 8. Volume manager

On the CLARiiON, metaLUNs offer much the same functionality, with a simpler interface for the host (many fewer LUNs to manage). There are subtleties in the difference between array-based management and host-based management. For example, concatenating or striping on the host allows 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 port throughput

A common and outdated technique for 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-size 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.)

This approach was critical in the days of SCSI-based storage systems, which were limited to about 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 9 shows three different plaid techniques. These are discussed in the next sections.

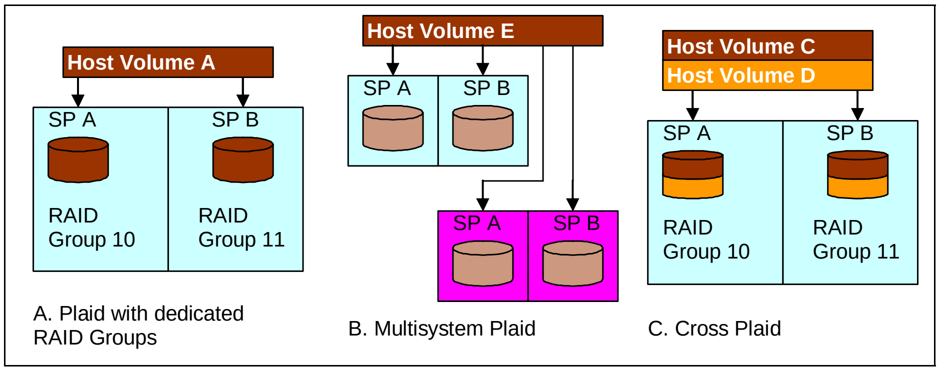


Figure 9. Plaid types

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. With dedicated disk groups, sequential access can be highly optimized down to the disk level. This approach is used when I/O sizes are large and access is (typically) sequential. See configuration A in Figure 9.

Unless I/O sizes from the application are very large (typically 2 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 concurrent access of all the LUNs in the volume stripe. To do that with a plaid, the host volume stripe must be filled with each I/O, and the stripe element on the host must be large enough to fill the LUN stripe underneath it. The result is a large volume manager stripe, which the application (or file system) must fill with each I/O for maximum efficiency. This may require some buffer size tuning or other adjustments to generate large writes.

Multisystem plaids

Plaids spanning storage systems (see configuration B in Figure 9) 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 (see configuration C in Figure 9) causing disk contention. Disk contention is not a problem for random access workloads, as they are going to cause a lot of disk seeks anyway. However, sequential I/O will be affected by multiple workloads on the drives.

Host virtual memory

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. The logical address space for an application is managed from a (limited) pool of real RAM and mirrored in full to a reserved space on disk. This area is called—depending on the OS—the page file or swap file. When a logical address is accessed, the VM reserves some RAM for it and loads the pages requested from the page file into the RAM. When an address is not in use, that part of an application’s memory may reside on disk (it is paged out). The VM operates on pages of RAM — much like the file system.

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 responsiveness of the system.

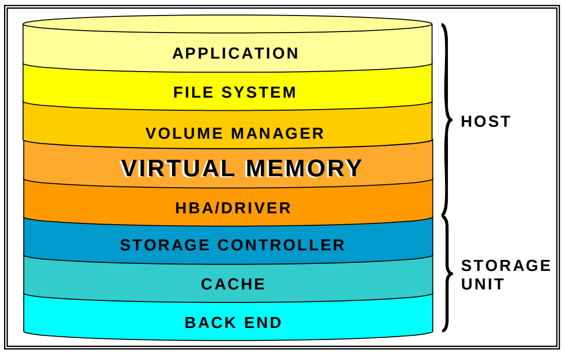


Figure 10. Virtual memory

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 for large memory usage). The first row shows throughput and bandwidth when storage-system write cache is enabled and the host VM is *not* tuned. The second row shows these parameters when storage-system write cache is disabled and the host VM *is* 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 the third row, 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. Be careful when installing RDBMS applications, which often take all available RAM. For this reason, the reserved file system buffers may have to be tuned before installing these applications.

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 4 Gb Fibre Channel HBA can sustain up to 380 MB/s throughput and about 40,000 IOPS.

If HA is required, you need at least two HBAs and a dual-SP storage-system configuration. Because of SP ownership, the LUNs to which the server is connected can be hosted by different SPs. This gives the host up to 760 MB/s in throughput (this configuration is illustrated in Figure 12). If a host requires more bandwidth to the storage system (and has the internal bandwidth to support it), HBAs can be added to reach the storage system's maximum bandwidth.

Ensure that the storage system can handle the desired bandwidth—check the performance papers available on EMC Powerlink. Also, deploy enough disk drives to deliver the target bandwidth.

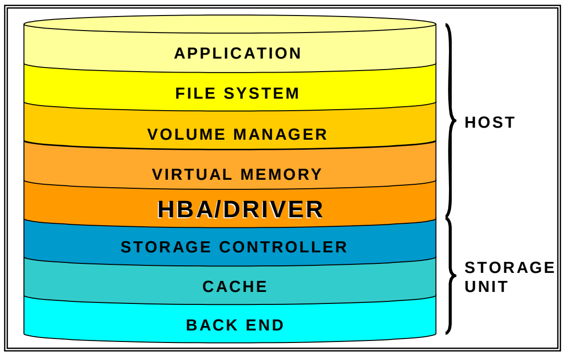


Figure 11. HBA/driver

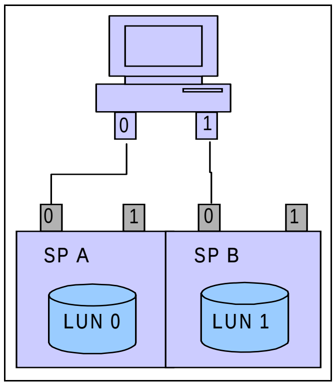


Figure 12. Using SP ownership HBA limitations

In Figure 12, a host with two HBAs uses the SP ownership model to address LUNs from both SPs. In this case, 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 required bandwidth is delivered to the host. HBA firmware, drivers, and storage-system hardware can all impose limits on the maximum I/O size to the storage system. For bandwidth planning, consult CLARiiON performance experts.

HBA settings

A common problem when setting up a SAN-based storage system is improper HBA queue depth settings. The HBA queue depth controls the maximum outstanding I/O sent from the HBA to the array, either on a total or per-LUN (depending on manufacturer) basis. Thus, the HBA is a likely place for a bottleneck that affects high-I/O-rate, transactional, performance.

EMC provides its technicians with a tool called the Procedure Generator, which includes proper settings for the HBA queue depth.

Failover and multipathing

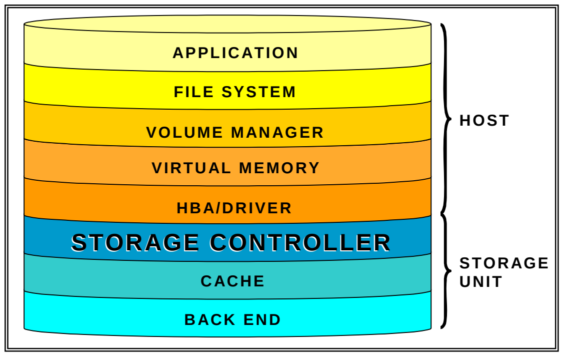
A highly available topology is part of the requirement for highly available systems. Another requirement is software that allows the host to connect to all of the storage system's ports—a technique referred to as multipathing. The benefits of multipathing are primarily the ability to fail over from a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

In FLARE® release 26, CLARiiON storage systems support Asymmetric Logical Unit Access (ALUA), which supports MPIO and the latest PVlinks from HP without any additional host software.

Failback returns the LUN to the original path, following the failure and subsequent correction of a path. This feature differentiates PowerPath from ALUA (which does not perform automatic failback). Another benefit of PowerPath is that given multiple active paths to a LUN, I/O load can be balanced across the storage processor ports and host HBAs to avoid bottlenecks. ALUA-based systems do not perform load balancing.

Storage controller effects

After leaving the HBA on the host, the I/O arrives at 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 for CLARiiON storage systems.



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

Figure 13. Storage controller

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?

CLARiiON storage processors

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

Storage processor enclosure-based systems

CLARiiON *storage processor enclosure* (SPE)-based systems are built around an enclosure that provides connectivity, cache, and cooling, but no disks. This design is found in the CX700 and all CX3 systems. Combined with UltraPoint™ and DAE-ATA disk enclosures, they allow a fully switched back end for the highest resiliency and greater diagnostics as compared to older, loop-based systems.

Disk processor enclosure-based systems

CLARiiON storage systems built around a *disk processor enclosure* (DPE) appear, from the front, to be a disk enclosure. However, in addition to space for 15 disk drives, the DPE provides connectivity, cache, and control in a single chassis. Within the first chassis, the controllers share power and cooling with the disks.

RAID levels

This paper has introduced these RAID concepts:

- Striping
- Mirroring
- Parity

This discussion addresses the impact these technologies have on performance.

Parity versus mirroring

When writing small I/Os to a parity (RAID 5, RAID 6 or RAID 3) LUN, the controller has to read, calculate, and write a parity segment for every data write operation. Two parity segments are written for RAID 6 LUNs. This increased write load influences the decision to use mirroring or parity-based RAID. (Refer to "Appendix: RAID 5" for details.) The total random I/O load on the drives for RAID 5 is:

Total disk IOPS = Read IOPS + 4* Write IOPS

In other words, to process a single *random* write, four disk I/Os must be performed. This is independent of the number of drives in the RAID group. RAID 6, since it has dual parity, has to perform six I/Os for a single random write.

For RAID 1 or RAID 1/0 it is:

Total disk IOPS = Read IOPS + 2* Write IOPS

Thus, the drives must do more work to absorb a random write on a parity RAID system than on a mirrored RAID system. When parity RAID is used, the load on the disk drives increases faster as host writes increase. One might think that writes on a RAID 5 system would be slower than a RAID 1/0 system; however, this is not necessarily so. If a system has write caching enabled and has not saturated the cache, the response time for a RAID 5 write is just as fast as a RAID 1/0 write, as they both return at cache speeds. But the RAID 1/0 system will service a greater random write load without stressing the cache.

Thus, RAID 1/0 is often suggested for systems experiencing high rates of random write I/O.

- 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.

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 write optimization

In some performance cases—writing large and sequential I/O—parity RAID 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. Less data must be transferred over the back end – and thus less SP and back-end resources are used for any particular I/O stream. This assumes that the parity RAID stripe can write full stripes and avoid the parity write penalty. (Parity is created on the fly from the full parity stripe held in memory.)

The technique CLARiiON uses is straightforward. With write cache enabled, the SP waits until sequential data for an entire stripe is cached before writing it. If uncached I/O is performed, and the I/O does not fill the stripe and it is not *aligned* to the RAID stripe, RAID 1/0 is a better choice.

The total I/O load on the drives when parity RAID is able to do full stripe writes is shown below. Note that we address the load in bandwidth (MB/s) because the disk IOPS rate changes; with cache coalescing and read prefetching. For example, the I/O sizes that the disks receive are often larger than those that were sent by the host:

Total disk MB/s = Read MB/s + Write MB/s * (1+ 1/number of disks in RAID group)

While the mirrored RAID set disk load is:

Total disk MB/s = Read MB/s + 2* Write MB/s

For a given number of drives, RAID 5 and RAID 1/0 are approximately equal in sequential read performance. RAID 5 typically offers some advantage in sequential write performance. RAID 6 has equal read performance to RAID 5, while a bit less in sequential writes.

RAID 5: Efficient, fast, flexible

RAID 5 offers the best mix of good random performance and great sequential performance at a low cost per GB. It provides the best capacity utilization, excellent protection against data and media failure, and the most flexibility for group sizes.

RAID 6: The highest integrity attainable

CLARiiON RAID 6 offers performance equivalent to or just below that of RAID 5, and capacity utilization between that of RAID 5 and RAID 1/0. CLARiiON RAID 6 is unique in the industry; the key advantage to this RAID type is protection from *two* drive failures, or a second failure during a group rebuild. It adds even more protection against media failures with multiple checksums and parity schemes, while offering true rotating parity for full *performance* utilization of all drives.

RAID 1/0: The random write leader

In RAID 1/0, the stripe segments are mirrored. A CLARiiON storage system can read from either the mirror or primary. Writes are committed to both. Compared to parity RAID, the lower load on the disk drives for random writes allows a system using RAID 1/0 to sustain a higher random write load, without saturating the cache.

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 significantly help 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.

RAID 1

RAID 1 has the same ability to absorb random writes that RAID 1/0 has. Since there is no striping, it does not distribute I/O as well as RAID 1 and thus is not extensively used. Some newcomers to CLARiiON will use RAID 1 with volume manager striping or metaLUNs to reproduce configurations they have used on other systems, but this is not the best use of a CLARiiON and should be avoided.

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, which helps ATA drives. The benefits of RAID 3 over traditional 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. This advantage has been reduced in the CX3-series CLARiiON systems, which use an extended rotating parity scheme that yields performance very close to that of RAID 3 with ATA drives.

In random write operations, the single parity drive is a bottleneck for parity reads and writes. For this reason, RAID 3 should be avoided when random writes are expected.

MetaLUNs

MetaLUNs are LUN objects created from multiple, smaller LUN objects. The purpose 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 and CX3 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 stored on 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 Storage* on EMC Powerlink. Operational semantics for metaLUNs can be found in *EMC CLARiiON MetaLUN - Concepts, Operations, and Management* on EMC Powerlink.

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.

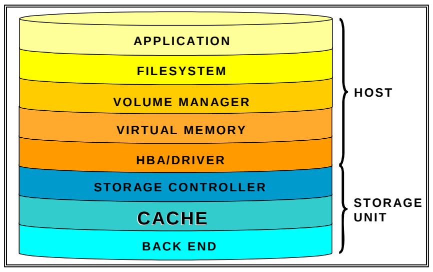


Figure 14. Cache

Cache structure

The CLARiiON cache is a region of the main SP memory that is separate from all other processes (such as working memory for the OS and replication software) and is contiguous. Memory can be partitioned online, though the cache must be deactivated in order to do so. The user partitions the cache into two regions: read cache, and write cache (a setting for RAID 3 buffering is held over in Navisphere from very old systems; that setting is no longer used).

Cache is organized into pages, which are the smallest unit of allocation. Incoming I/O is placed in the page, and if they are smaller than the page, and contiguous, multiple I/O may share the same page.

The vault

The first five drives of any CLARiiON hold a special area known as the *vault.* The cache is dumped (written at high speed) into the vault if a failure threatens the system's ability to protect cache. Thus vault drives have less usable space than other drives, and any drive bound in a RAID group that uses a vault disk is similarly reduced in capacity.

Topics regarding the vault and performance impact are covered in *EMC CLARiiON Best Practices for Fibre Channel Storage* on EMC Powerlink.

Cache 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. Instead, more read hits are likely from the write cache. When raw devices are in use, the typical application is an RDBMS, which will do its own read buffering.

As a result, the requirement for read cache is modest, usually a fraction of that allocated for write cache.

Read prefetch

Prefetch is beneficial in two ways. Firstly, the disk gets much better performance when doing sequential I/O. Storage systems leverage this characteristic by reading ahead many blocks when sequential access is detected, a process known as *prefetch*. Secondly, since the data is in cache before the host requests it, the response time experienced by the host is excellent.

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—100 MB to 250 MB per storage processor is typical.

Write cache

The main advantage of write cache is it decouples the response time of writes from disk speed and RAID effects. If the storage system can safely cache the I/O in memory and return status to the host, a random write can be completed (from the host's perspective) in far less time than it takes to put the data on disk.

Sequential writes offer many opportunities for optimization, and the write cache is the key for executing those optimizations. Many smaller writes can be coalesced into fewer large transfers to the disk drives. If enough data is in cache, an entire parity stripe can be written out.

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 come 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 advantage to a larger cache is its ability to absorb bursts, and the ability to achieve read hits form the cached data.

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 that would otherwise be benign.

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? The answer is that there are several costs associated with all write cache mirroring schemes:

- There is 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, because 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 to maximize bandwidth without sacrificing random-access operations.

Activating the write cache

Navisphere provides controls for caching. 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:

- At least one 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 – see note below).
- Both SPs must be present and functional.
- Both power supplies must be present and functional in the DPE/SPE. In SPE models, both DAE 0 power supplies must be functional. The CX3-10, -20, -40 can have one power supply fail and still maintain cache.
- For CX series, all fan packs must be present and functional in the DPE/SPE. CX3 series systems will maintain the write cache with one fan faulted.
- DAE 0 must have two nonfaulted link control cards (LCCs).

There is a caching option in the storage system **Properties** dialog box that you select with the **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 *not* selected, write caching remains active when a single vault disk fails.

More information on write cache operation can be found on EMC Powerlink.

Cache settings

The CLARiiON RAID engine cache is one of the most tunable of any storage system. Yet, the only setting that the user must enter is allocation (size of read and write caches); standard settings work fine for most applications. But, for applications with atypical performance profiles, 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)
- 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.

For systems performing mostly large I/O or sequential I/O, a larger cache page size can help overall system performance. However, for most systems, where access is mixed, the default 8 KB size is the best.

Prefetch settings

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

Variable prefetch is the default 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 **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 **Prefetch Disable Size** is another limit designed to prevent prefetch operations from monopolizing the drives. I/Os of the disable size and larger will not cause a prefetch operation.

The prefetch idle count limits the execution of prefetch for a very busy LUN. The idle count is also used in cache flushing; changing this parameter should be done only at the prompting of EMC service personnel.

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, flushing pages of "idle" LUNs. (An idle LUN is one that has had no I/O for two seconds or longer.) Idle flushing is the primary way that cache pages get flushed.
- 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 occurs when the cache reaches 100 percent capacity. A full cache significantly increases response time.

Figure 15 illustrates how 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.

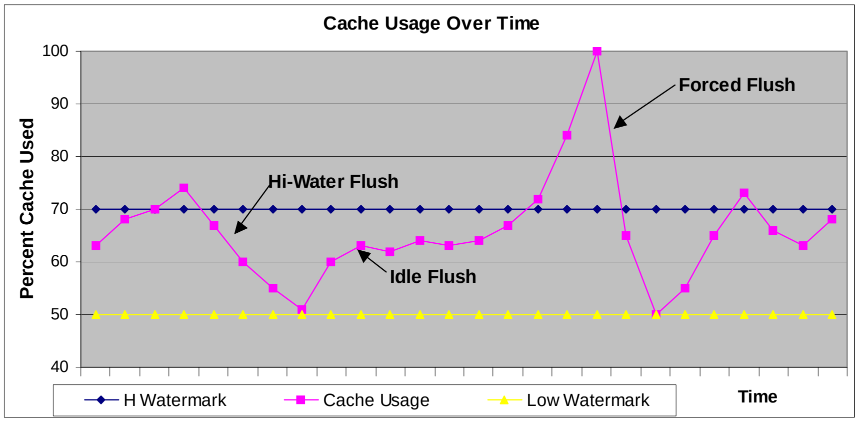


Figure 15. Depiction of watermarks with steady-state and flush behavior

Using the watermarks, the user can tune the amount of cache overhead and reserve cache pages to match a client's workload. For example, lowering both watermarks will increase the cache's "reserve" of free pages, allowing the system to absorb a burst of application writes. Generally, the goal is to avoid forced flushes, because when a host hits a forced flush, the write must wait for some disk operations before the I/O can proceed. The best practice for setting the low watermark is 20% below the high watermark.

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. (This function is sometimes referred to as "write-through" in storage literature.) 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 write-aside size parameter sets the largest I/O that will be write cached. The default is 2048 blocks; I/O of *more than* 1 MB will bypass cache.

Refer to *EMC CLARiiON Best Practices for Fibre Channel Storage* on EMC Powerlink 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.

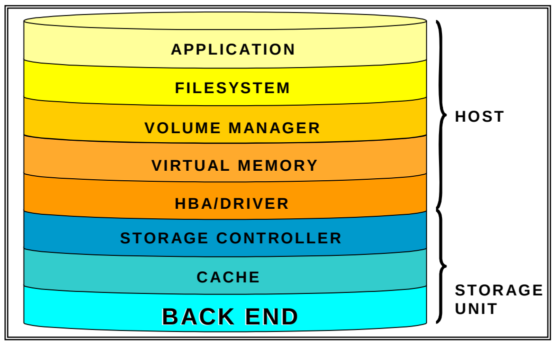


Figure 16. Back end

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?

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 128 LUNs. LUNs are striped evenly across all the disks of a RAID group. Figure 17 shows a graphical illustration 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.

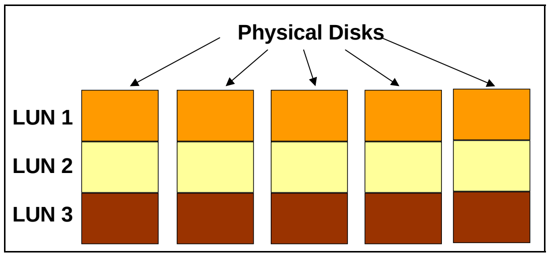


Figure 17. Distribution of LUNs on a partitioned CLARiiON RAID group

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.

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.

Figure 18 shows a simple 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.

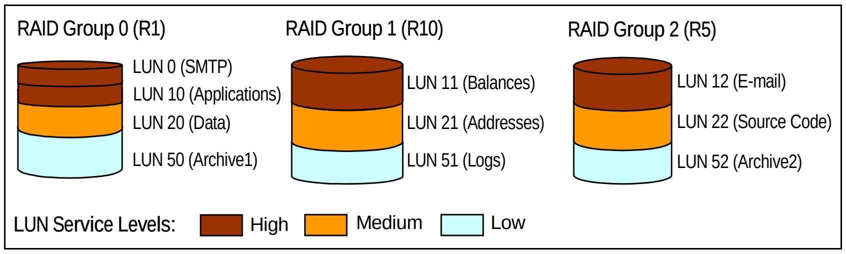


Figure 18. Sample LUN distribution

MetaLUN striping can be used to distribute load over many drives without the need to "custom fit" applications and LUNs. In complex environments, this is the best way to get the most performance from the available drives. Please see *EMC CLARiiON Best Practices for Fibre Channel Storage* on EMC Powerlink for details.

Keeping drives busy

A fundamental maxim in performance is that IOPS and bandwidth performance increase 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 (concurrency) in order to keep many drives working in parallel.

Systems that by nature have good concurrency are databases, email servers, and large enterprise applications, as from Oracle, SAP, and others. Some applications do not have good concurrency and have to be tuned for good performance; media streaming applications are notable for that characteristic.

Minimizing disk contention

Disk contention is the term used to describe the condition in which different host I/O threads interact at the disk level, causing a high level of seeks relative to the number of I/Os involved. Multiple applications accessing the drives is not in itself a problem: Random access is random, whether from one application or 10. As long as the aggregate load does not exceed the drive's performance potential, you do not need to worry about random-on-random contention.

However, the mixing of file systems expecting sequential I/O with file systems expecting random I/O in the same RAID group is problematic. It 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. This is especially true with ATA drives. Contention for spindles causes seeks, and extracts a cost in increased response time.

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.

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 6 effective disk count is two less than the number of drives. Six-disk RAID 6 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 at the direction of EMC. 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 / 6-disk RAID 6	256 KB
6-disk RAID 5	320 KB
9-disk RAID 5/ 10-disk RAID 6	512 KB
8-disk RAID 1/0	256 KB
16-disk RAID 1/0	512 KB

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. This is usually at or near the limit of allowed disk drives for CLARiiON systems.

For larger or sequential I/O, loop bandwidth becomes the limiting factor. Due to the very high transfer rates of Fibre Channel disk drives, a fairly modest number of drives can exceed the bandwidth of the disk loops.

As part of an EMC Professional Services engagement, EMC performance experts can help design a system to maximize performance potential.

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 result in less latency with reads.

The rotational speed affects cache flushing for writes that 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, and all writes return at cache speeds.

ATA drives

The design focus for ATA drives is low cost and reasonably good performance, but they lack some of the refinements of Fibre Channel drives. 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 enables data that was once aged to tape to remain online.

CLARiiON systems use several techniques to maximize the bandwidth performance of ATA drives, particularly when used with RAID 3, RAID 5, or RAID 6.

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 the host losing access to data, and while preserving data already stored.

The scope for discussing reliability is not as great as performance or 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. The CX3 series storage systems' systemic reliability is rated at 99.999 percent uptime (exact reliability measurements depend on the model and components used). Data availability can be increased by following the redundancy guidelines presented in this section.

Redundancy is achieved through the intrinsic qualities of the storage system itself and through selection of RAID type, front-end connectivity and, to a lesser extent, back end. High-availability attach, for example, is a way to achieve redundancy.

Highly available attach

The *EMC CLARiiON Open Systems Configuration Guide* (on EMC Powerlink) outlines the attach methodologies and equipment that are supported. For each topology in the "Configurations Diagrams" section, the guide explains whether it is completely, partially, or non-highly available. See Figure 19, which is excerpted from that document.

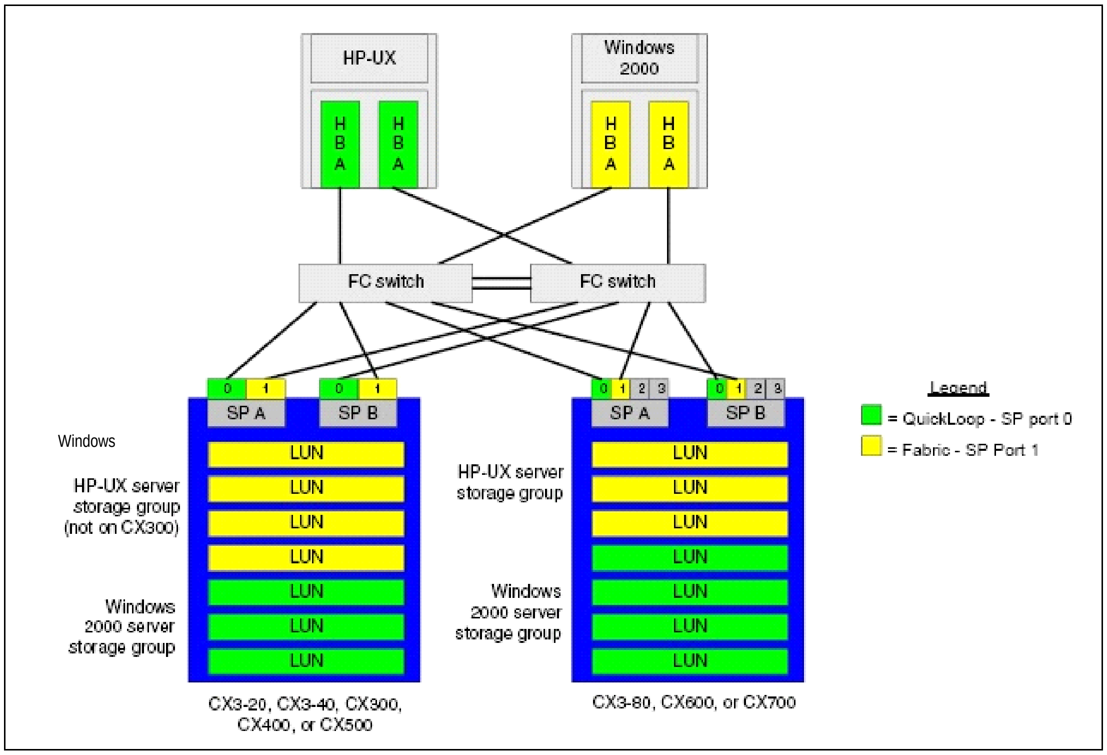

HP-UX
Windows 2000
HBA
HBA
HBA
HBA
FC switch
FC switch
0
1
SP A
SP B
2
3
Legend
= QuickLoop - SP port 0
= Fabric - SP Port 1
Windows
HP-UX server storage group (not on CX300)
Windows 2000 server storage group
HP-UX server storage group
Windows 2000 server storage group
LUN
CX3-20, CX3-40, CX300, CX400, or CX500
CX3-80, CX600, or CX700

Figure 19. High-availability configurations

The Windows and HP-UX hosts 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 access any data from one switch.

A host with only one HBA can survive an SP outage (such as during a firmware upgrade) because the switches are linked by means of an interswitch link (ISL). It can not access storage in the case of an HBA failure.

You must use two HBAs (with or without a fabric) or a single HBA and a switch, with paths to both SPs, to ensure access to data in the case of SP or back-end failures. Additionally, some type of host failover system (PowerPath, MPIO, PVlinks and so forth) is required. In the unlikely case of a back-end cable or link failure, the access to the LUN will be from the peer SP, so failover must be designed into the system to cover both front-end and back-end failures.

RAID level considerations

The RAID level determines the amount of redundancy at the disk level. Redundancy allows the RAID driver to recover data in the case of a media failure, and in the case of a total disk failure.

RAID 0 offers no data redundancy. RAID 1 and RAID 1/0 offer 100 percent redundancy; that is, up to half of the disks can fail or suffer media failures 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. RAID 6 can withstand two disk failures—either concurrent disk failures, or a media failure during a rebuild of a previous disk failure.

Striping across disks is usually desired for performance and capacity reasons. RAID 0, RAID 3, RAID 5, RAID 6, and RAID 1/0 are all striped RAID implementations. RAID 0's lack of redundancy limits its use. RAID 3 is usually used only when large and sequential I/O is guaranteed. Most customers choose RAID 1/0 or RAID 5, so we will examine these options in detail. We will also discuss the new RAID 6 option.

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 disk group. 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. Users needing many drives for capacity or performance are encouraged to use the metaLUN feature, and use sets of RAID groups.

The highest redundancy with RAID 5 can be achieved by striping across all the disk buses (3+1) in a CX700 or CX3-80.

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).

When using a system with two or more back-end buses (per SP), if a RAID 1/0 group is bound across two disk buses (with primaries on one bus and mirrors on the other), the user can achieve the highest level of redundancy.

RAID 6

RAID 6 can withstand two disk failures and still store and return data reliably. This also applies to media failures: If a disk fails to read a block, the block is reconstructed from the redundant data located elsewhere in the RAID stripe. CLARiiON RAID 6 excels at overcoming combinations of disk and media failures, exceeding even RAID 1/0 in this function. Advanced checksums and diagnostics have been built into CLARiiON RAID 6, resulting in the most redundant RAID scheme in the industry.

RAID 6 offers many times the redundancy of RAID 1/0, with slightly lower random IOPS than RAID 5. The sequential performance per disk is as good as RAID 5 but, since it is transmitting more parity data to the drives, overall system bandwidth is lower than with RAID 5. RAID 6, then, should be used where redundancy requirements outweigh performance requirements.

Using the cache

The CLARiiON cache is a mirrored write-back cache, which means that data is stored in memory, copied to the peer, then the command **complete** is sent to the host. For relational database management systems that instruct the user to disable cache, or to use “durable storage,” the CLARiiON cache is considered durable storage and need not be disabled for these applications.

The **HA Vault** setting is one of the few modifications the user can make to redundancy of the CLARiiON that is not in the attach design. By default, the cache will be disabled if the **HA Vault** setting is selected and a vault drive fails. Clearing this box will benefit performance greatly in these cases.

What is the tradeoff? The **HA Vault** setting protects against a triple failure: a vault drive failure followed closely by a power failure, which itself is accompanied by a *second* vault drive failure—either while the vault is being dumped to disk, or on restart of the storage system. This is a very unlikely event, but the user should be familiar with the conditions before making the decision to turn off the **HA Vault** setting.

Conclusion

The CLARiiON is a very flexible system that will make the most of any disk technology used with it – whether that's FC or SATA. Out of the box, the default settings provide excellent performance for most uses. The key to maximizing performance is understanding the entire system, from the application down to the disk. Only with knowledge of the application profile and the transformations made to it by the intervening layers can the user predictably change performance by altering CLARiiON settings.

This paper is intended as an introduction to CLARiiON performance. The next step in understanding how to get the most from a CLARiiON storage system is the *EMC CLARiiON Best Practices for Fibre Channel Storage* white paper on EMC Powerlink.

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 large I/O requests (larger than 64 KB) at relatively low IOPS.

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 a 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 — Coalescing is the process of gathering multiple, contiguous I/O requests into one large request.

crossing (disk) — If a back-end I/O is not contained in a single stripe element, a *disk crossing* is incurred, which can signal degraded performance for small-block, random I/O. Note that an I/O larger than the stripe element size (128 blocks = 64 KB) will incur a stripe crossing; this is normal.

crossing (stripe) — If a back-end I/O is not contained in an entire stripe, a *stripe crossing* is incurred, which takes more time (due to 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.

dirty page — A write cache page with valid (not yet flushed) data.

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 standby power supply.

disk crossing — When, in a striped RAID disk set, an I/O spans two drives. This is expected with I/O larger than the **stripe segment**, but with smaller I/O it is due to the **alignment** being off and degrades performance (by making to disks busy instead of one).

disk processor enclosure (DPE) — The chassis element containing the CX300/CX500 SPs, first 15 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 standby power supply (SPS). See also **standby 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.

metadata — Any data outside of application-addressable space, typically used by the partitioning process or file system during housekeeping. The information stored by applications is referred to as *user data*.

metaLUN— A virtual LUN object built by striping or concatenating multiple LUN objects. A CLARiiON base feature.

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 less than 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.

queue — Set of operations waiting to be serviced, for any service center (port, disk, SP and so forth).

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**.

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 a typical midrange system is about 0.99995, which translates into expected downtime of 30 minutes per year.

request size — In a file system, the size of the block actually read from (or written to) the disk; this often differs from the application's request.

response time — The cumulative time for an I/O completion as measured from the host (or upstream service center; for example, LUN waiting for a disk operation).

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.

scaling – The ability of a system to increase performance as disk drives are added.

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

service time — The time for a service center (disk, SP and so forth) to process an I/O, once the I/O is dequeued.

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/stripe 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 the group multiplied by the stripe element size equals 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 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 standby power supply.

standby 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 `chglun` command.

Appendix: 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.

RAID 5 stores a mathematical construct known as parity, which is derived from the data in the stripe. Parity is used to reconstruct data lost from one drive and is saved on a block-by-block basis in the parity segment. Each 512-byte block in the parity segment holds the parity calculation for the set of blocks *in that same position* on the data drive stripe segments. So, for example, parity block 100 represents the parity of the data blocks located at LBA 100. This is known as "horizontal" parity, as the parity is calculated across the stripe.

An advantage of block-based horizontal parity is that the CLARiiON need update only the parity blocks that correspond to the data blocks changed on the data segment. So, for example, if the host writes 4 KB, we update only 4 KB of parity.

The parity write penalty

The requirement to compute parity for each stripe of data written to 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 20 shows the steps the storage system takes to execute an I/O to a single disk in a parity RAID scheme.

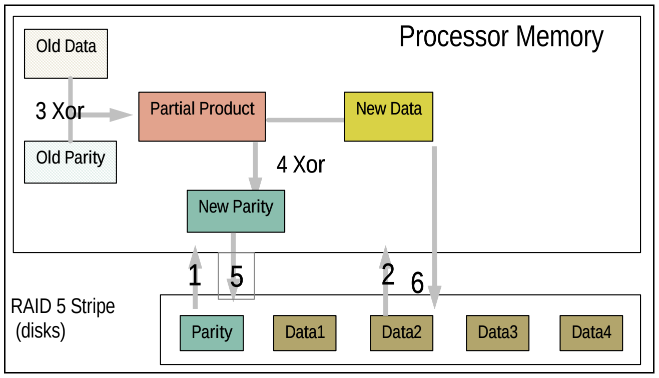


Figure 20. RAID 5 write penalty

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 from its disk into memory.
3. In memory, an *XOR* (Boolean exclusive or) is performed on the old data and old parity to create a partial product.
4. An XOR is performed on the new data and the partial product to create the new parity for the stripe.
5. The new parity is written to the disk.
6. The new data is written out.

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

The parity write penalty and RAID 6

RAID 6 uses standard horizontal parity, just the same as RAID 5. In order to reconstruct data if two data drives are lost, another type of parity called “diagonal” parity is used. Diagonal parity is parity calculated across increasing locations as the stripe is traversed. (This is a complex operation and this paper will not attempt to describe the details of the CLARiiON RAID 6 implementation.) Diagonal parity is stored in the second parity segment in the stripe. That is why RAID 6 requires two extra drives per stripe, versus only one additional drive for RAID 3 and RAID 5.

In RAID 6, two parity elements are read and two parity elements are written. Thus, there are a total of six disk I/Os to update a stripe with a small, random write. Note, that since the CLARiiON “diagonal” parity calculated the diagonals *within the disk block*, no additional blocks need be read from the data drives.

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. This is illustrated in Figure 21.

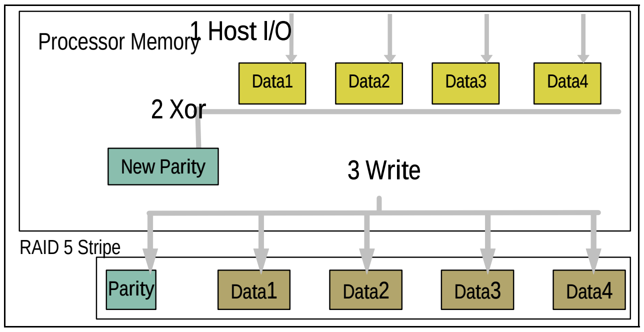


Figure 21. RAID 5 MR3 write

The steps are:

1. The stripe is received from the host as a large I/O or many sequential I/Os.

2. The entire stripe is XORed in memory to produce a new parity segment.

3. The 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 requires less data to be written, which is advantageous for high-bandwidth applications.

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 of equal capacity can provide more optimal performance than a RAID 1/0 LUN.