Engineering White Paper

EMC CLARiiON Best Practices for Fibre Channel Storage

CLARiiON Firmware Release 13 Update (non-Confidential)

Abstract

This paper presents the best performance and redundancy practices for implementing CLARiiON storage systems. The paper concentrates on system settings, RAID types, metaLUNs, and the impact of host I/O patterns on performance.

Published 3/16/2004

Part Number H519.4

Table of Contents

Executive Summary

This white paper presents the best practices for achieving high performance and reliability as they pertain to CLARiiON® Fibre Channel storage systems.

The paper also explains how to properly size a CLARiiON storage system for performance and capacity.

What's New in Release 13

Release 13 introduced three new CLARiiON storage systems. So, in this paper, you will see references to the new models:

- CX300
- CX500
- CX700

There is a performance enhancement specific to these systems. This affects the performance sizing for these systems, and the *Sizing Example* (on page 35) has details.

References to the FC series products are being reduced.

Release 13 also introduced changes to RAID 3, which had significant impact on ATA drives. Refer to the section *ATA Drives and RAID Levels* on page 30 for details.

The information regarding MR3 optimization has been expanded and clarified. Refer to *CLARiiON RAID 5 Stripe Optimizations* on page 27.

Intended Audience

The intended audience for this white paper is technical personnel who require guidance with the best approaches to implementing CLARiiON Fibre Channel storage. An understanding of the basics of mirrored and parity RAID is required, as is a knowledge of CLARiiON fundamentals.

Introduction

This paper outlines factors affecting the performance and availability of CLARiiON Fibre Channel, block-level storage on both storage area networks (SANs) and direct-attached storage (DAS). The various components in the system that affect *performance* and *availability* are addressed in succession—from the application, to the host, to the storage system itself.

Fundamental concepts that support the conclusions in this white paper[1] can be found in the following paper, posted on Powerlink™ with general availability:

EMC CLARiiON Fibre Channel Storage Fundamentals

This paper follows the same format as the fundamentals paper to enable the latter to be used as a reference.

[1] To keep this paper concise, fundamental explanations were removed.

Considerations for Performance

How important is performance tuning? Using RAID 5 in groups of five to nine drives and using default settings, the CLARiiON Fibre Channel storage systems deliver excellent performance—this is how EMC benchmarks its CLARiiON systems in the Performance Engineering lab.

The default settings for a CLARiiON storage system are designed to address perhaps 80 percent of the workloads encountered in the real world. However, the other 20 percent of workloads require tuning to make the best use of the storage system.

Why use groups of five to nine drives? There is nothing magic about this configuration, nor are there special optimizations for the configuration. However, RAID 5 groups of this size are relatively efficient in their parity usage, and rebuild in a reasonable time. Smaller groups have a higher parity cost, while larger groups take longer to rebuild.

These and other considerations are explained in detail in this section.

Defining Performance

The following terms are used throughout this paper. If you are not familiar with them, please review *EMC CLARiiON Fibre Channel Storage Fundamentals*.

- Throughput
- Bandwidth
- Response time
- Saturation
- Random
- Sequential
- Prefetch
- Request size
- Burstiness
- Write-aside

Application Design

The application design determines much of the behavior of the system. Some application characteristics discussed in this section are:

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

Application Efficiency

The first best practice to follow is application tuning. Efficiency in this case refers to two factors:

- Requesting no more data than is necessary for good application performance
- Requesting information in a manner that leverages storage-system behavior

No amount of storage-system tuning will make up for a badly designed application.

Factors to consider are:

- Optimizing for sequential or random access
- Reads versus writes
- I/O size
- Patterns in data access: bursty or steady
- Application buffering, threading, and asynchronous I/O

Optimizing Sequential or Random I/O

Typically, tuning for sequential access pays off in improved throughput and lower response times, as the storage system behaves more efficiently with sequential I/O—especially with RAID 5. This does not imply that entire files or tables should be read in order to access several individual records: data that is required should be read. If a file or table will be cached and referenced multiple times, then a full scan or read is efficient. If not, individual records should be read.

Note that seeking backwards on a file is inefficient and should be avoided.

Reads versus Writes

In general, writes are more expensive to execute than reads. The drives take longer to service a write operation, and redundant RAID schemes require more back-end transfers to perform a write than a read—this is especially pronounced in parity-based RAID.

To balance the relatively high cost of writes, the CLARiiON storage system uses a write cache, which insulates the host from the physical latency of the drives. There is a small latency for mirroring the cache, and the write cache can smooth out bursts of writes over time. But the higher the write load, the harder the drives must work, and if the drives are very busy, response times for reads will suffer and the cache may saturate.

Thus, it is in the application's best interest to minimize writes. For example, storing multiple copies of data is inefficient. Use a single copy and use pointers or references. Normalize your databases before committing them to production.

I/O Size

For random access applications, such as account updates, OLTP, etc., transaction rates and response times are better with smaller I/O sizes. Normalization of the database results in smaller records, though more joins will be required. This may result in marginally more read operations, but the accesses should be smaller, and the write load may be reduced, which, with RAID 5, can pay off significantly.

Typically, bandwidth increases with I/O size. Configuring the host to send large I/Os makes sense when moving large amounts of data. However, on a CLARiiON storage system, both write caching and prefetch are configured to be bypassed when I/O reaches a certain size. The decision to use a large request or break it into smaller sequential requests depends on the application and its interaction with the cache. These interactions are discussed in *The RAID Engine Cache* on page 22.

Generally, when rereads of the data are not expected, use large (greater than 1 MB) I/O requests when writing large files to a RAID 3 or RAID 5 LUN. Also, large I/O sizes are effective for reads of any large dataset (including backups).

Use smaller I/O sizes on sequential writes when data must be cached for reread. An interesting example is an RDBMS TEMP table. The TEMP data is written and then reread; if the writes bypass the cache, they take longer than if cached. Also, subsequent rereads have to go to disk (no possibility of a cache hit). Using smaller requests is faster: writes hit the write cache and thus return more quickly, and the reread can be serviced from data still in the write cache—much faster than going to disk.

Information regarding the per-LUN parameters can be found in the discussion on *write-aside* and *prefetch* settings in *The RAID Engine Cache* on page 22.

File system settings can affect I/O sizes as well, as described later in the *Host File System Impact* section.

Temporal Patterns and Peak Activities

The operational design of the application—how it is used, when it is used, when backups occur—should be made with some consideration of storage-system load, particularly if the storage system is shared between applications. Backups and batch processes should be scheduled so that peak times do not overlap. This results in more predictable performance.

Application Buffering, Threading, and Asynchronous I/O

Use application buffers as much as possible: more RAM on the server will help. Application buffering is typically the most intelligent and has the lowest response time. Read the *Best Practices* white papers from EMC for Oracle, Microsoft Exchange, and SQL Server to determine how to leverage these applications.

Generally, for OLTP and messaging applications (anything with a high rate of random access I/O), more threads are better. The storage system gains high throughput when there is a lot of concurrency. Asynchronous I/O is effective as well.

For bandwidth, single-threaded applications are rarely able to make use of more than four effective drives, unless request sizes are very large (greater than 2 MB).

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 System Buffering and Coalescing

Similar to caching on the storage system, buffering is one of the primary means by which file systems enhance performance.

Buffering

In most cases, file system buffering should be maximized, as it reduces load on the storage system. There are, however, some exceptions to this rule.

Traditionally, applications that do their own buffering avoid, or work around, file system buffering. This is because of the assumption that the application can buffer more intelligently. Also, by avoiding the file system's coalescing, the application has more control over the I/O response time. However, as RAM in 64-bit servers grows to 32 GB and beyond, it becomes possible to buffer entire file systems. This can reduce response times for reads—which are buffered—tremendously. (Writes should use a write-through feature to ensure persistence of committed data.)

Coalescing

File system coalescing can assist in getting high bandwidth from the storage system. In most sequential-access operations, file system coalescing should be maximized by using the `maximum contiguous` and `maximum physical` file system settings (when available).

However, coalescing does not always help. Take the example of multiple threads writing small I/Os. Each thread waits for the response from the storage system before it moves on to its next task. But if the file system coalesces these requests into one large request, *all* the threads are now waiting for a single large I/O, which has a much longer response time than the smaller I/Os. This can impair RDBMS performance (by coalescing writes) and is another reason why RDBMSs are often deployed on raw devices.

Minimum I/O Size: The File System Request Size

Since the storage system can handle smaller I/Os in less time than larger ones, it is advantageous in OLTP-type applications to use the smallest request size possible. File systems that offer configurable block sizes (down to 2 KB) offer an advantage to OLTP applications. Raw partitions, whose request sizes are not limited by a file system, also help by reducing the I/O size to the minimum supported by the drives (512 bytes).

Maximum I/O Size

If the goal is to move large amounts of data quickly, then a larger I/O size will help. Large I/O—those that are two or more times the size of the CLARiiON RAID stripe—result in sequential operations at the drive level, which is extremely efficient. Large I/O sizes are also critical in getting good bandwidth from host-based multiLUN stripes, as well as CLARiiON metaLUNs.

File System Fragmentation

Avoid fragmentation and defragment on a regular basis. Note that if NTFS file systems are formatted at anything but the default extent size, they *cannot* be defragmented with most tools: the API does not allow it. Some tools can work around this, but at a higher risk. Performing a *file-level* copy (to another LUN or by executing a backup and restore of the file system) is an effective approach to defragmenting.

File System Alignment

File system misalignment affects performance in two ways:

- Misalignment causes disk crossings of I/O, which would otherwise require only one drive to service them.
- Misalignment makes it hard to stripe-align uncached, large writes.

The first case is more commonly encountered. Even if the disk operations are buffered by cache, the effect can be detrimental, as it will slow flushing from cache. Random reads, which by nature require disk access, are also affected, both directly (waiting for two drives in order to return data) and indirectly (making the disks busier than they need to be affects response times of all I/O).

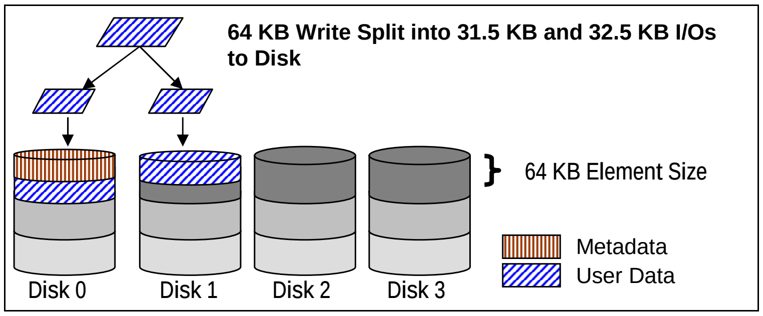

Figure 1. Effect of Misalignment with a 63-Block Metadata Area

Disk crossings with small I/O can be significant with some host types due to the issues discussed in the following sections.

Alignment on Intel Architecture Systems

This issue affects Linux systems using Intel architecture, as well as Microsoft Windows NT, 2000, and 2003 systems.

The `fdisk` utility, as well as the Windows Disk Manager, places a Master Boot Record (MBR) on every logical device (i.e., drive letter). The Master Boot Record itself is 16 KB, but an entry in this record specifies *hidden sectors* on the drive. The placement of the MBR at the beginning of a logical device causes subsequent data structures to become misaligned with respect to the CLARiiON RAID stripe.

In Windows NT, the misalignment is 32 blocks; in Windows 2000 and 2003, it is by 63 blocks. On Linux systems, the hidden sector amount depends on the boot loader and/or disk manager software used.

In any case, the result causes misalignment for some larger I/O. For Windows 2000 and 2003, the boundary is on an odd value—31.5 KB—so almost all I/O is subject to disk crossings.

Adapting to Metadata

The two approaches for adapting to this phenomenon are using the Navisphere® LUN *Alignment Offset* and using a partition utility. *Use only one approach*, not both, for any particular LUN. Also, when setting up a metaLUN, only the base LUN requires stripe alignment.

Avoid using the LUN offset method if SnapView™, SAN Copy™, or MirrorView™ will be used on this LUN. Instead, use a host-based disk utility.

The LUN Offset

The LUN offset method aligns the partition on the *stripe* boundary. The downside of the LUN Alignment Offset is that by aligning the partition, it *misaligns* the I/O requests made by any software that operates on the raw LUN. MirrorView and SnapView operate on the raw LUN, and they will incur disk crossings if the LUN has been offset.

The offset is specified in Navisphere when the LUN is bound, and the value is designated in 512-byte blocks. Set the LUN Alignment Offset value to the amount by which your data is offset. For instance, on a Windows 2003 system, use 63 blocks as the offset value. FLARE™ will adjust the stripe so that user data starts at the beginning of the stripe.

Disk Partition Adjustment for Windows Systems

In Windows NT, 2000, and 2003 systems, the utility `diskpar.exe`—part of the Windows Resource Kit—can be used to set the starting offset of a partition. You must do this before data is written to the LUN, as `diskpar` rewrites the partition map: all existing data on the LUN will be made unavailable.

For random-access operations, set the starting offset in `diskpar` to the stripe element size used to bind the LUN (typically 128 blocks). For high-bandwidth applications, set the starting offset equal to the LUN stripe size.[2] You can obtain the drive number by using Disk Manager.

To prepare, check the existing starting offset using the `–i` switch. Issue `diskpar -i x` (where *x* is the drive number) from the command line:

```
C:\>diskpar -i 0
---- Drive 0 Geometry Information ---- <deleted for brevity>
---- Drive Partition 0 Information ----
StatringOffset = 32256
PartitionLength = 40007729664
HiddenSectors = 63
. . .
```

Note the `HiddenSectors` value. This is the amount by which the partition is offset.

1. If disk X is a raw drive, skip to step 3. If disk X has data that you don't want to lose, back up that data.

2. Delete all partitions on disk X, making it a raw disk.

3. Issue `diskpar -s x` (where *x* is the drive number) from the command line.

4. Enter the new starting offset (in sectors) and the partition length (in MB). This step writes the new offset to the MBR for that drive and creates the partition. After you enter the starting offset and partition size, the MBR is modified and the new partition information appears.

5. Issue `diskpar -i x` (where *x* is the drive number) at the command prompt to review the information on the newly created partition.

The `diskpar.exe` method is preferable to the LUN Alignment Offset method for LUNs that will have a snapshot, BCV, or MirrorView image made of them. It is preferred for SAN Copy sources and targets as well.

[2] Refer to *Stripes and the Stripe Element Size* in the *CLARiiON Fundamentals* white paper for a definition of stripe size.

Disk Partition Adjustment for Linux Systems

In Linux, align the partition table before data is written to the LUN, as the partition map will be rewritten and all data on the LUN destroyed. In the following example, the LUN is mapped to `/dev/emcpowerah`, and the LUN stripe element size is 128 blocks. Arguments for the `fdisk` utility are shown below:

```
fdisk /dev/emcpowerah
x   # expert mode
b   # adjust starting block number
1   # choose partition 1
128 # set it to 128, our stripe element size
w   # write the new partition
```

This method is preferable to the LUN Alignment Offset method for LUNs that will have a snapshot, BCV, or MirrorView image made of them. It is preferred for SAN Copy sources and targets as well.

Volume Managers

The primary performance impact of Volume Managers is the manner in which CLARiiON LUNs are striped (known as a plaid or stripe on stripe).

Avoid using *host-based RAID implementations* that require parity (for example, RAID 3, RAID 5). This consumes host resources for a service (parity protection) that is better handled by the storage system.

Figure 2 shows three different plaid techniques that are discussed in this section.

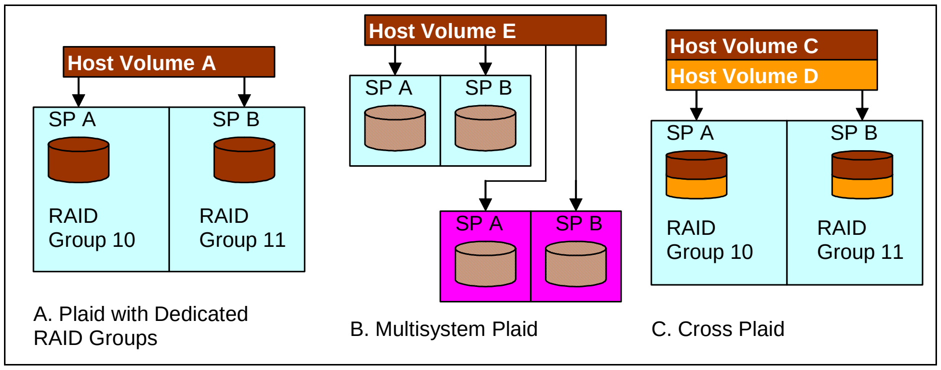

Figure 2. Plaid Types

Plaids for High Bandwidth

Plaids get used for high-bandwidth applications because access for a large volume can be split across more than one processor or system. However, care must be taken or throughput will be less than with a dedicated RAID group. Also, accessing multiple storage-system ports makes no sense unless you are using multiple HBAs on the host. For each active path to a CLARiiON storage processor, you will need an HBA on the host. For example, if you plan to drive I/O simultaneously through two ports on SP A and two ports on SP B, you will need four HBAs on the host to achieve maximum bandwidth.

When working with very large I/O sizes using RAID 3 or RAID 5, a plaid can be effective *if the I/O size is large enough*. This means that any single host I/O will result in full-stripe requests to all drives in the Volume Manager stripe.

For bandwidth, it is best if the disk groups used are dedicated (no other LUNs contending for access). As shown in Figure 2 (configuration A), the LUNs making up a plaid can be in separate SPs in order to distribute load across storage-system resources and to maximize port bandwidth. Note that the host uses two HBAs, each of which is zoned to a single port on each SP.

Set the host stripe segment size equal to or to a multiple of the stripe size of the CLARiiON LUNs, upon which the stripe is built. For example:

- With four LUNs striped, each with a LUN stripe size of 128 KB, the application should be able to issue a 512 KB or larger I/O (4 * 128 = 512).
- With eight LUNs striped, each with a LUN stripe size of 256 KB, the application should be able to issue a 2 MB or larger I/O (8 * 256 = 2048).

The latter example points out the challenge in performing high bandwidth to a device that is distributed across many drives: the I/O size must be very large to use the drives efficiently.

Plaids are useful when applications are single-threaded (e.g., reading a single large file). The Volume Manager will create a worker thread for each stripe segment, thus increasing the level of concurrent access to the storage system. As long as the application can send very large I/O sizes to the Volume Manager, the Volume Manager can effectively work many disks in parallel.

Spanning storage systems, as shown in Figure 2 (configuration B), is suggested *only* when file system sizes and bandwidth requirements warrant such a design. For example, a 30 TB geological information system (GIS) database that requires write bandwidth in excess of 500 MB/s is a candidate for a multisystem plaid. Note that an NDU or any storage-system fault—such as deactivation of the write cache due to a component failure on one storage system—affects the entire file system.

Plaids and OLTP

In OLTP applications, which are hard to analyze and which suffer from hot spots, plaids are an effective tactic to distribute I/O across many spindles. An application that can keep many drives busy benefits from a high-drive count.

Note that some Volume Managers recommend small host stripes (16 KB to 64 KB). This is unnecessary and suboptimal for CLARiiON LUNs, which use a striped RAID type. Also, backup and restore times may suffer. The Volume Manager *stripe element* should be set to the CLARiiON *stripe size* (typically 128 KB or 256 KB).

The primary cost of a plaid for OLTP purposes is that most users end up with a cross plaid.

The Cross Plaid

Disks—and thus disk groups—are getting larger, and users usually end up with RAID groups sharing host volume stripes (a *cross plaid*—see Figure 2, configuration C). For instance, instead of four nine-drive RAID groups, each with a dedicated LUN (each of the four LUNs presented as a volume to the host), a user may end up with four RAID groups, four LUNs on each group, and four volume groups on the host. The volume groups would be made up of a LUN from each RAID group.

The rationale for this design is that any burst of random activity to any one-volume group is distributed over 36 drives, rather than nine. The downside is that determining interactions between volumes is extremely difficult. However, a cross plaid may be effective when:

- I/O sizes are small in size (8 KB or less) and randomly accessed.
- The volumes are subjected to bursts at different times of the day, not at the same time.

Plaid *Don'ts*

- Don't stripe multiple LUNs from the same RAID group together. This only causes large disk seeks. If you need to combine multiple LUNs from one disk group, *concatenate* contiguous LUNs—do not use striping.
- Don't make the host stripe element less than the CLARiiON RAID stripe size.
- Don't plaid together LUNs from RAID groups with differing RAID types, stripe sizes, or radically different drive counts.

Host HBA Effects

The topology used for host attach depends on the goals of the system. High availability demands dual HBAs and dual paths to storage. The performance impact of dual-pathing lies mainly in the ability of the administrator to balance load across storage-system resources.

Keep in mind HBA and driver behaviors when tuning a storage system. The EMC E-Lab™ provides suggested settings for drives and firmware, and these suggestions should be followed.

HBA Limitations

HBA firmware, drivers, and hardware can all impose limits on the maximum I/O size to the storage system. Some typical limitations for the widely used Emulex HBAs are:

- Max I/O size: 512 KB on Windows NT, with the 1.27a(x) driver
- Max I/O size: 1.3 MB on Windows 2000/2003, with the 5.2.21.3 driver

When planning a high-bandwidth application, take HBA maximums into account. The HBA firmware, the HBA driver version used, and the operating system of the host can all affect the maximum I/O size the HBA can handle. Also, the HBAs and firmware have bandwidth limits. Table 1 shows some observed limitations based on HBAs and host bus configurations (PCI and PCI-X).

Table 1. Sample of HBA Observed Maximum Bandwidth Results

HBA	Bus	Storage System Tested	Write Bandwidth, MB/s	Read Bandwidth, MB/s
LP8000	PCI	FC4700	87	85
LP9002	PCI	CX600	110	170
LP9802	PCI	CX600	130	190
QLA 2340	PCI	CX600	130	190
LP9002	PCIX	CX600	150	170
LP9802	PCIX	CX600	190	190
QLA 2340	PCIX	CX600	194	190

Linux I/O Fragmenting

The Linux kernel is not yet fully optimized for Fibre Channel attach. Linux fragments requests from the application or file system into chunks. Chunks for raw devices are 512 bytes; for file systems, they are 4 KB. These chunks may (or may not) be reassembled before the host dispatches to the storage system.

The problem was fixed in the `varyio` patch, which was included in the Red Hat 2.4.9-e.12 AS/ES 2.1 errata release. This patch was intended to reduce fragmentation to no smaller than the Linux page size (4 KB). However, observed behavior in later Red Hat patches indicates a continuing fragmentation, mainly in writes.

With Linux, avoid concurrent access of multiple large files on LUNs: this causes many requests from different threads to use different pseudo-devices to the same underlying device. The interference reduces the coalescing of writes. It is better to use many smaller LUNs, each with a single large file.

PowerPath

If available on the OS, PowerPath® should always be used, whether for a single-attach system through a switch (which allows an NDU to remain non-disruptive) or in a fully redundant system.

In addition to basic failover, PowerPath allows the host to connect to a LUN via multiple storage processor ports—a technique referred to as multipathing. PowerPath optimizes multipathed LUNs with load-balancing algorithms. PowerPath offers several load-balancing algorithms, but the default—ClarOpt—is recommended. ClarOpt adjusts for number of bytes transferred, as well as queue depth.

Hosts connected to most CLARiiON models can benefit from multipathing, as all but the CX200 have dual host ports per SP. DAS multipathing requires at least four HBAs; practical SAN multipathing requires two HBAs, each zoned to more than one SP port. The advantages of multipathing are:

- Failover from port to port on the same SP, which is faster than a failover to the peer SP
- Load balancing across SP ports and host HBAs
- Higher bandwidth attach from host to storage system (assuming the host has as many HBAs as paths used)

While PowerPath offers load balancing across all available active paths, this comes at some cost:

- Every active and passive path from the host requires an initiator record, and there is a finite number of initiators per system (32 per port on the CX300, CX400, and CX600, 64 per port on the CX500 and CX700).
- Active paths increase time to fail over in some situations (PowerPath tries several paths before trespassing).

Because of these factors, active paths should be limited, via zoning, to two storage-system ports per HBA for each storage system to which the host is attached. The exception is in environments where bursts from other hosts that share storage-system ports are unpredictable and severe. In this case, four storage-system ports per HBA can be used.

MetaLUNs

MetaLUNs are a standard feature of all CX series storage systems. There are several topics that address when and how to use a metaLUN. Some are administrative—not performance-related—but to be thorough, they are discussed in this section.

When to Use a MetaLUN Instead of a Volume Manager

On a CLARiiON storage system, metaLUNs are implemented as a layer above the RAID engine, and are functionally similar to the application of a Volume Manager on a host. However, there are some important distinctions between metaLUNs and a Volume Manager.

Single SCSI Target versus Many

To create a Volume Manager stripe, all the component LUNs must be made accessible to the host. MetaLUNs require only a single SCSI LUN to be mapped to the host; the multiple LUNs, which make up the metaLUN, are not seen by the host. This benefits the administrator in several situations:

- In hosts with limited LUNs available due to OS limits
- In hosts where adding LUNs causes a renumbering of SCSI devices; often a kernel rebuild is necessary to clean up the device entries

In these cases, using a metaLUN instead of a Volume Manager simplifies administration on the host.

No Volume Manager or Incompatibility with PowerPath

Not all OSs have Volume Manager support. On some, use of a Volume Manager may preclude the use of PowerPath. In these cases, a metaLUN allows LUN virtualization that is compatible with PowerPath.

Microsoft Windows Server 2000/2003 clusters using Microsoft Cluster Services (MSCS) cannot make use of dynamic disks. MetaLUNs are a solution for providing expandable, striped, and concatenated volumes for these systems.

Storage Processor Bandwidth

An important distinction between a Volume Manager volume and a metaLUN is that a metaLUN is addressed entirely by one storage processor on one CLARiiON storage system. If very high bandwidth is required for a volume, a Volume Manager is still the best approach, as the volume can be built from LUNs on different SPs and storage systems. A Volume Manager allows the user to access storage at an aggregate bandwidth of many storage processors.[3]

Creation of Volume Manager Threads and Concurrency

As pointed out in the *Plaids for High Bandwidth* section on page 11, the use of a host-striped volume has the effect of multithreading large requests (those that consist of more than one volume stripe segment). This increases concurrency to the storage system and is a technique that is effective in speeding up single-threaded operations, such as reading or writing a large file.

A metaLUN does not have this effect. Because the multiplexing of the component LUNs is done on the storage system, there is no multithreading effect. If such an effect is needed, use a Volume Manager.

Replication of the Volume

If the volume is to be replicated on the storage system using SnapView, MirrorView, or SAN Copy, a metaLUN will simplify replication enormously.

Volume Access Sharing

When a striped or concatenated volume must allow shared access between hosts, and a Volume Manager will not permit shared access, a metaLUN can be used. The metaLUN is placed in both hosts' storage groups.

Very Large I/O sizes

While it is not confirmed by testing yet, there is evidence that unless tuned properly, metaLUNs will be less effective than Volume Manager stripes at executing I/Os that span multiple metaLUN stripe segments. Follow the guidelines for large I/O sizes described in the next section *MetaLUN Usage and Recommendations*.

MetaLUN Usage and Recommendations

The three types of metaLUNs are: striped, concatenated, and hybrid. This section presents general recommendations. For those who want more detail, the following subsections address strategies for creating metaLUNs and the relative merits of each type.

When to Use MetaLUNs

Given the previous Volume Manager discussion, you should use a metaLUN In the following cases:

- When multiple-LUN storage consolidation is necessary (large file systems or many drives per volume are needed)
- When LUN expansion is a requirement (you can wait until the expansion is required to modify a LUN into a metaLUN)

Do *not* use metaLUNs if two datasets have radically different access profiles; the performance advantage of having them on separate RAID groups may outweigh the convenience of having them on a single device. Mixing of sequential workloads and random workloads is not optimal.

You can control the following factors when setting up a metaLUN: component LUN type, metaLUN type, and stripe multiplier.

Component LUN Type

The LUN type you bind for inclusion in a metaLUN should reflect the I/O pattern expected for the metaLUN. For instance, the recommendations made in this document for different RAID types apply (refer to the *RAID Levels and Performance* section on page 21).

[3] Many try but few achieve such high bandwidth; it requires *very* large I/O sizes to keep many disk drives busy!

When binding component LUNs, follow these rules:

- Always use the default stripe element size (128 blocks) when binding LUNs for use in metaLUNs.[4]
- Always activate read and write cache.
- Set the write-aside size for component LUNs to 2048. (Write-aside is covered in *The RAID Engine Cache* on page 22.)
- Avoid using LUNs from RAID 5 groups of less than four drives (in other words, use 3+1 or larger).
- Do not use the LUN offset to adjust for stripe alignment. MetaLUNs have their own offset value.

MetaLUN Type

In general, you should use striped metaLUN components wherever possible, as they yield the most predictable performance. Concatenation of a single LUN to a metaLUN is intended for convenience; this may be appropriate for expanding a volume that is not performance-sensitive.

Hybrid metaLUNs combine concatenation with striping. A striped metaLUN can be expanded by *concatenating another striped component.* This preserves the predictable performance of a striped component, and allows you to expand a striped metaLUN without restriping existing data (performance is impacted while the restriping operation is under way). See Figure 3.

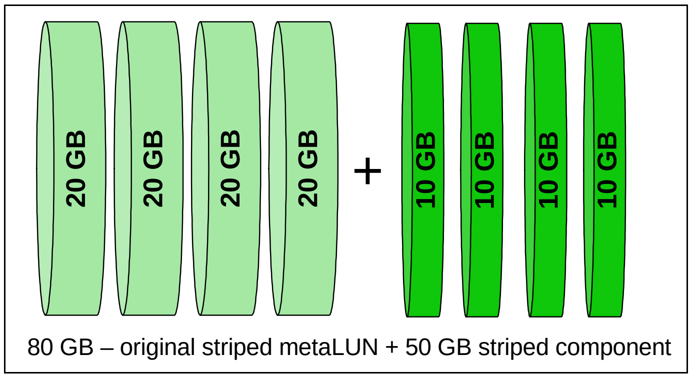

Figure 3. Hybrid-Striped MetaLUN

Ideally, the LUNs in the concatenated stripe set are distributed over different RAID groups of the same RAID type and with the same number of drives as the original striped component. The most direct way to achieve this is to use the same RAID groups as the base component. This approach is illustrated next.

MetaLUN Stripe Multiplier

The stripe multiplier determines the metaLUN stripe segment size:

*stripe multiplier * base LUN stripe size = metaLUN stripe segment size*

The metaLUN stripe segment size is the *largest I/O any component LUN will receive*. The following guidelines will cause a maximum I/O size of about 1 MB in most cases, which is ideal. See Figure 4.

- For RAID 1, RAID 1/0, RAID 5, and RAID 3 use **four**.
- For RAID 0, set the stripe multiplier to **two**.

Since you have bound the LUNs with the default LUN stripe element size of 128 blocks (64 KB), this ensures the metaLUN stripe segment will be large enough to do full-stripe writes to the component LUNs, even after the component LUNs have been expanded.

[4] For some reason, many people feel a need to "tune" their CLARiiON storage systems and change the stripe element size. It has been recommended for some years to *use the default stripe element size unless instructed to do otherwise by CLARiiON Performance Engineering.*

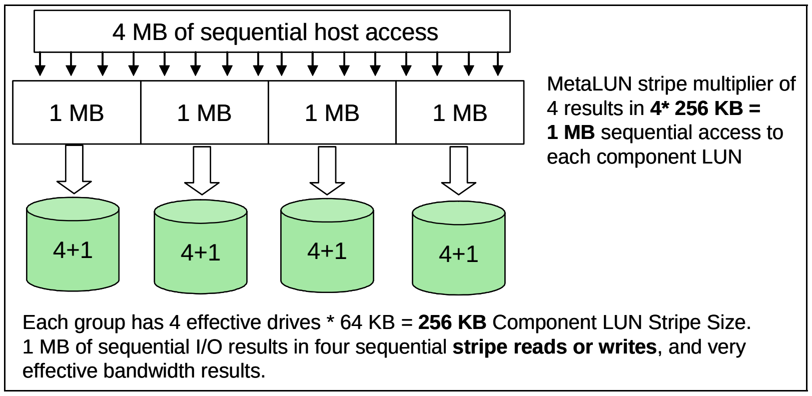
4 MB of sequential host access
1 MB
1 MB
1 MB
1 MB
MetaLUN stripe multiplier of 4 results in **4* 256 KB = 1 MB** sequential access to each component LUN
4+1
4+1
4+1
4+1
Each group has 4 effective drives * 64 KB = **256 KB** Component LUN Stripe Size.
1 MB of sequential I/O results in four sequential **stripe reads or writes**, and very effective bandwidth results.

Figure 4. MetaLUN Stripe Multipliers and Bandwidth

MetaLUN Alignment Offset

If you plan to use SnapView or MirrorView with your metaLUN, leave the metaLUN **Alignment Offset** value at zero. Use disk utilities to adjust for partition offsets.

MetaLUNs and ATA Drives

ATA drives are not a solution for busy random I/O access. This section concentrates strictly on using ATA drives for high bandwidth applications.

Keeping RAID groups small is part of the metaLUN strategy. This makes sense for ATA drives, as small groups have shorter rebuild times than large groups. However, be aware that a metaLUN is impacted by a single group's rebuild, and the impact of rebuild on ATA stripe sets is lengthy. For the purposes of data availability, it may be best in many environments to avoid using metaLUNs with ATA drives unless dynamic expansion is a requirement.

CLI Example: Creating a MetaLUN

In the following example code, we create a striped metaLUN. There is no **create** command for metaLUNs; you create a metaLUN by expanding an existing FLARE LUN. All the LUNs designated are FLARE LUNs of the same RAID type and the same capacity. LUN 30 will become the base—the new metaLUN will keep 30 as its identifier.

```
metalun -expand -base 30 -lus 31 32 33 -name P1H00 -elszm 4 -type S
```

Note: The expansion type is set to `S`, for striped, and the element size (4) is selected because the LUNs are built on RAID 5 groups with five drives.

MetaLUN Expansion Strategies

There are several strategies for using metaLUNs for a long-term expansion plan. To develop a strategy, you must identify your goals. The goals of the approach presented in the following section are:

- Distribution of localized bursts of otherwise random data over many disk drives
- Good sequential/bandwidth performance
- Efficient use of capacity
- Flexible expansion of devices

These goals apply to the majority of metaLUN users.

Expansion Model Initial Configuration

The rules for the initial setup of this solution are illustrated in Figure 5. The rules are:

- Deploy required drives for initial deployment capacity.
- Create modest sized RAID groups:
  - For RAID 1/0, use four or six drives.

    - For RAID 5 or RAID 3, use five drives.
- Organize the RAID groups into *sets* of four to eight groups. (Use more groups per set if very high rates of random I/O are required.)
- For each metaLUN, determine the RAID group set to which it will belong.
- Define the component LUN size for each planned metaLUN by dividing metaLUN size by the number of RAID groups in its RAID group set.
- Create a component LUN for each metaLUN from *each* RAID group in its set.
- Form metaLUNs from LUNs distributed across all the RAID groups in their respective sets.

See Figure 5 for an example of a set of metaLUNs and their RAID group set.

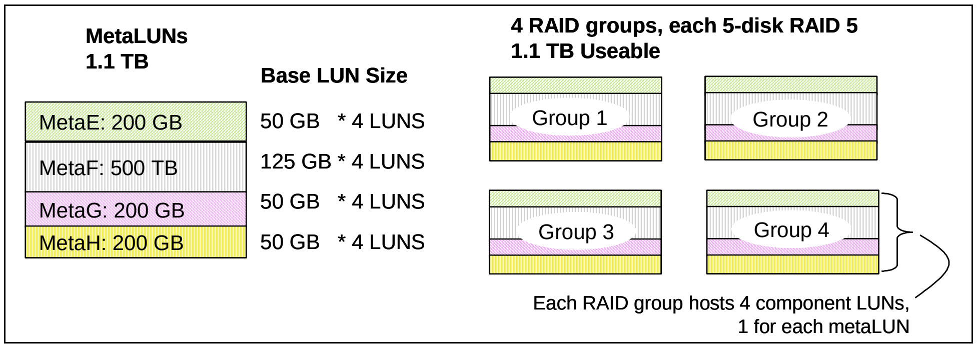

Figure 5. Intial Distribution of Storage for MetaLUNs

Note that in Figure 5, each metaLUN consists of one LUN per RAID group. Thus, each LUN's load is evenly distributed across all RAID groups in that set. However, these metaLUNs are fenced off from data access to other RAID group sets.

Why use RAID group sets? If we do not allow a metaLUN to extend outside of its set, we can determine a level of fencing, controlling interactions at the drive level. For instance, one RAID group set may be for a large number of file servers, while another is used for RDBMS data tables—and an ordinary pair of RAID 1 groups may be used as the RDBMS log devices. See Figure 6.

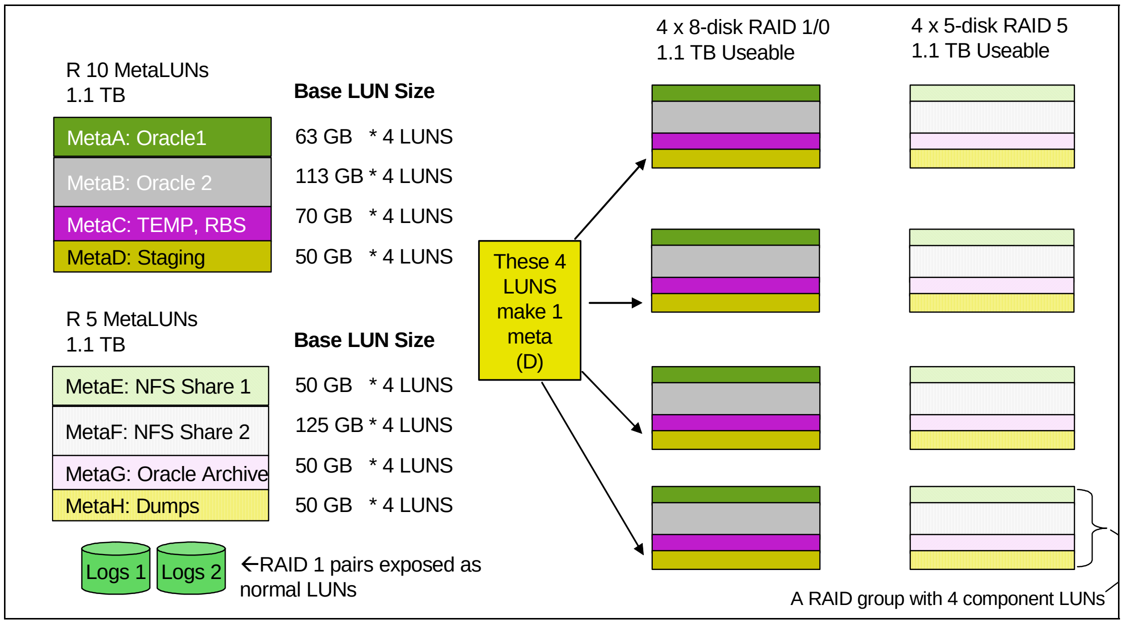

Figure 6. Example of Data Fencing with RAID Group Sets and MetaLUNs

In the example shown in Figure 6, access to the NFS share metaLUNs will not interfere with the Oracle servers' access to their data tables or to their logs.

Expansion Model Expansion Procedure

The next step is to set up the strategy for expansion. The goals for expansion are:

- Maintain distribution of data across many drives.
- Use capacity efficiently.

The approach to achieve these goals is:

- When capacity is *anticipated* for a metaLUN, add drives to existing RAID groups in the set.
- Bind expansion LUNs on the RAID groups of the metaLUN's set.
- Add expansion LUNs to metaLUNs as a new striped component.

MetaLUN Expansion Example

The approach used in this example adheres to the original goals for the metaLUN configuration—I/O distribution across all disk drives.

In the first step, the IS department determines that capacity usage for Meta A is over their watermark—85 percent—and notifies the users of that metaLUN. IS receives a request for an additional 80 GB by the end of the week. The operator of this system adds two drives to each RAID group in the set on which metaLUN A resides. RAID group expansion can be executed at night, during a nonbusy period. See Figure 7.

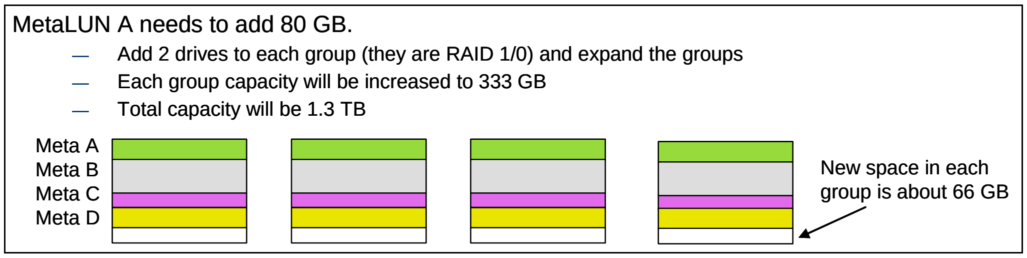
MetaLUN A needs to add 80 GB.
— Add 2 drives to each group (they are RAID 1/0) and expand the groups
— Each group capacity will be increased to 333 GB
— Total capacity will be 1.3 TB
Meta A
Meta B
Meta C
Meta D
New space in each group is about 66 GB

Figure 7. Expansion for MetaLUNs: Step 1

The next step is to bind a LUN in each RAID group of the metaLUN's set. The amount by which they must expand is 80 GB, and there are four RAID groups in the metaLUN set, so 80/4 = 20. A 20 GB LUN must be bound on each RAID group in the set.

The last step is to expand the metaLUN using the four new bound LUNs. The operator designates the LUNs to be added and sets the expansion as *concatenated*. Because the expansion LUNs are all the same size, Navisphere concatenates a new *striped* component to the metaLUN, consisting of these LUNs. See Figure 8.

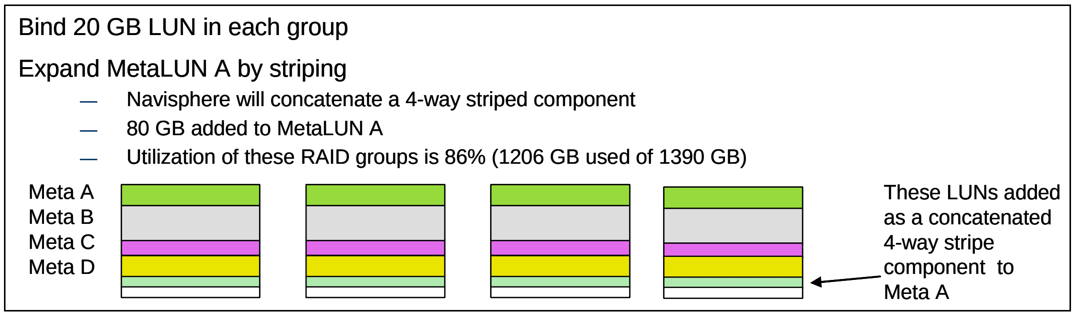
Bind 20 GB LUN in each group
Expand MetaLUN A by striping
— Navisphere will concatenate a 4-way striped component
— 80 GB added to MetaLUN A
— Utilization of these RAID groups is 86% (1206 GB used of 1390 GB)
Meta A
Meta B
Meta C
Meta D
These LUNs added as a concatenated 4-way stripe component to Meta A

Figure 8. Expansion for MetaLUNs: Step 2

The following is an example CLI command for expanding a metaLUN by concatenating a new striped component. The metaLUN identifier is 30. FLARE LUNs 34, 35, 36, and 37 all have the same RAID type and capacity:

```
metalun -expand -base 30 -lus 34 35 36 37 -type C
```

Note: The expansion type is set to `C`, for concatenated. Navisphere will stripe the LUNs together in a new component to be added to the existing metaLUN, metaLUN 30.

MetaLUN Base LUN Stacking

When creating multiple metaLUNs from a set of RAID groups as in the example above, rotate the RAID group in which you locate the base LUN for each metaLUN. This distributes the "hot edge" of a database, file system, or even of a backup process among the disk drives. See Figure 9. Each color stripe denotes a different metaLUN; the base LUN for each meta is on a different RAID group.

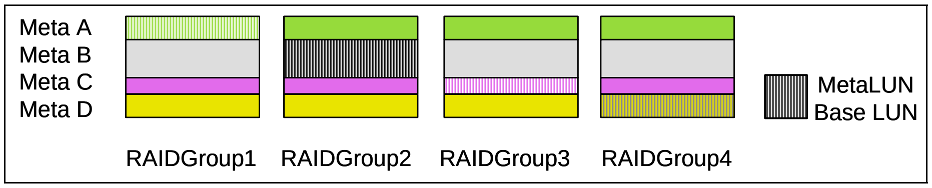

Figure 9. MetaLUN Base LUN Stacking

Storage Controller Effects

This section guides the user in matching the required performance with the appropriate CLARiiON hardware. This section also discusses how to set the LUN and system attributes, and provides details on how best to use the different RAID types.

CLARiiON Storage Processors

The current CLARiiON lineup consists of the models below. Some notes regarding their advantages over the older CX models are included.

Furthermore, all the new CX systems (CX300, CX500, and CX700) share improvements in write cache architecture, which will result in higher random write throughput.

CX700

The CX700 shares the storage processor enclosure (SPE) design with the CX600. The CX700 offers a faster chipset and memory subsystem as compared to the CX600, as well as double the disk bandwidth (it has four redundant disk buses on the back end). Bandwidth and IOPS performance of the CX700 is greatest, disk for disk, than any other CLARiiON system. The CX700 represents the best choice for the highest performance and greatest scalability.

CX500

The CX500 uses a small form-factor SP that fits in the link control card (LCC) slot of a disk-array enclosure (DAE). However, the CX500 SP offers dual CPU (versus the single CPU CX400 SP) and a chipset faster than that in the CX400.

In steady random I/O environments, the CX500 performs slightly below the CX700 up to its maximum complement of 120 drives. The CX500 has a smaller write cache than the CX700, and thus will not absorb as large a burst of host writes. The CX500 is well-balanced performer. The CX500 provides much improved bandwidth over the CX400, offering near "wire speed" with large, sequential I/O and Fibre Channel drives.

CX300
The CX300 shares similar hardware with the older CX400, but it has half the number of disk ports. It performs as well as the CX400 up to its limit of 60 drives for random access. However, due to its single back-end disk bus, it performs closer to the CX200 for bandwidth.

RAID Levels and Performance

CLARiiON storage systems often use RAID 5 for data protection and performance. RAID 1/0 is used when appropriate, but the decision to use RAID 1/0 does not always depend on performance. Refer to the *CLARiiON Block Storage Fundamentals* white paper to learn how RAID 5 delivers high performance.

When to Use RAID 5
RAID 5 is favored for OLTP, messaging, data mining, medium-performance media serving, and RDBMS implementations in which the DBA is effectively using read-ahead and write-behind. If the host OS and HBA are capable of greater than 64 KB transfers, RAID 5 is a compelling choice.

These application types are ideal for RAID 5:

- Any operations where the write load does not cause cache saturation
- A DSS database in which access is sequential (performing statistical analysis on sales records)
- Any RDBMS tablespace where record size is larger than 64 KB and access is random (personnel records with binary content, such as photographs)
- RDBMS log activity
- Messaging applications
- Video/media
- When cost concerns outweigh performance concerns

When to Use RAID 1/0
RAID 1/0 can outperform RAID 5 in workloads that use very small, random, and *write-intensive* I/O. Some examples of random, small I/O workloads are:

- High-transaction-rate OLTP
- Large messaging installations
- Real-time data/brokerage records
- RDBMS data tables containing small records that are updated frequently (account balances)

If random write performance is the paramount concern, RAID 1/0 should be used for these applications.

When to Use RAID 1
RAID 1 is used when a dedicated file system is required, and the storage needs are small enough to make a RAID 1/0 LUN too costly. Often, an RDBMS transaction log is implemented on a RAID 1 LUN.

However, RAID 1 is not striped, which means it does not deliver the performance advantages of a striped RAID level. RAID 1 does not handle multiple streams of random access well.

Furthermore, RAID 1 is not suitable for writing very large I/O sizes. If the host requests are larger than 128 KB in size, latency at the disk level results. (A striped RAID level breaks these larger requests up as it stripes across the disks.)

When to Use RAID 3
RAID 3 is a specialty solution. Only five-disk and nine-disk RAID group sizes are valid for CLARiiON RAID 3. The problem target for RAID 3 is large, sequential data.

With Release 13, RAID 3 LUNs can now use write cache. The restrictions previously made for RAID 3—single writer, perfect alignment with the RAID stripe—are no longer necessary, as the write cache will align the data. RAID 3 is now more effective with multiple writing streams, smaller I/O sizes (such as 64 KB) and unaligned data.

RAID 3 is particularly effective with ATA drives, bringing their bandwidth performance up to Fibre Channel levels. The RAID 3 changes are valid for all CX-series storage systems.

The RAID Engine Cache

This section discusses the proper cache page size to use, how to use prefetch size, where to set watermarks, and other techniques, such as SP balancing.

Cache Size and Performance

Refer to the *CLARiiON Block Storage Fundamentals* white paper for general information about the impact of cache size on performance.

Read Cache

For systems with modest prefetch requirements (about 80 percent of installed systems), 50 MB to 100 MB of read cache per SP is sufficient.

For heavy sequential read environments (requests greater than 64 KB and sequential reads from many LUNs expected over 300 MB/s), use up to 250 MB of read cache. For extremely heavy sequential read environments (120 or more drives reading in parallel), up to 1 GB of read cache can be effectively used by the CX600.

Write Cache

Set the read cache as just explained, and then allocate the remaining memory to write cache.

How to Select the Right CX600 Model

The CX600 comes in two models—the 4 GB and 8 GB versions. Approximately 500 MB per storage processor (1 GB total) is used for system processes. Write cache must be mirrored. Thus, the 4 GB version allows a maximum of approximately 1.5 GB for write cache, assuming all available memory is dedicated to write cache. The 8 GB model allows allocating the maximum to write cache (3 GB) and a large read cache as well.

The 8 GB model benefits dynamic high-bandwidth applications, where writes and reads are concurrent or equally distributed over time. The larger cache also benefits large databases, as checkpoints cause a burst of writes, and efficient handling of a checkpoint is critical to ongoing database operations. Heavy use of BCVs (business continuance volumes) and mirrors requires more write cache as well, especially if RAID 5 or slower disk-drive technologies are used for the BCV or mirror LUN. Refer to the *ATA Drives* section on page 29 for more details on using ATA drives.

The CX700 is available in an 8 GB cache model only.

Scripting Cache Sizes

In many environments, production workloads vary or are much different than night time/off-hours batch workloads. Using the Navisphere CLI, the CLARiiON cache allocation can be scripted to adjust to periodic workload shifts. Scripts can be included in scheduled jobs to automate adjustments to workloads. Refer to the *Navisphere CLI Command Reference* for the syntax for adjusting cache allocation.

Cache Settings

The cache parameters for CLARiiON storage systems *all have default settings that will benefit most users.*

Caches On or Off

Most workloads benefit from both read and write cache; the default for both is **on**.

To save a very small amount of service time (a fraction of a millisecond to check the caches when a read arrives), turn off read caching on LUNs that will not benefit from it. For example, LUNs with very random read environments (no sequential access) will not benefit from read cache. Use Navisphere CLI scripts to turn on read cache for LUNs when preparing to perform backups.

Write caching is beneficial in all but the most extreme write environments, and deactivation of write cache is best done using the per-LUN *write-aside* setting (refer to the *Write-Aside Size* section on page 23).

Page Size

In cases where I/O size is very stable, you gain some benefit by setting the cache page size to the request size seen by the storage system—the file system block size or, if raw partitions are used, application block size.

In environments with varying I/O sizes, the 8 KB page size is optimal.

Be careful when applying a 2 KB cache page size. Sequential writes to RAID groups with misaligned stripes and RAID 5 groups with more than eight drives may be affected. Refer to the *CLARiiON RAID 5 Stripe Optimizations* section on page 27 for details.

The "HA Vault" Option and Write Cache Behavior

Starting in Release 12, a new option in the global cache settings was made available. The *HA Cache Vault* option, found on the **Cache** page of the storage-system properties dialog box, is on (selected) by default. The default is for classic CLARiiON cache vault behavior, as outlined in the *Fibre Channel Best Practices Guide*.

Several failures (outlined in the Best Practices paper) cause the write cache to disable and dump its contents to the vault. One type of failure is that of a vault drive. If the user *clears* the **HA Cache Vault** selection, then a vault disk failure will not cause write cache to disable. Since a disabled write cache significantly impacts host I/O, it is desirable to keep the write cache active as much as possible.

Clearing this selection exposes the user to the possibility of data loss in a triple-fault situation: If a drive fails, then power is lost, and then *another drive fails during the dump*, it is not possible to dump the cache to the vault. The user must make the decision based on the relative merit versus risk.

Prefetch Settings

The default setting for prefetch (**Variable**, with segment and multiplier set to **4**) causes efficient cache behavior for most workloads.

You should consider increasing the prefetch multiplier when *both* of the following conditions apply:

- I/O request sizes are small (less than 32 KB).
- Heavy sequential reads are expected.

Decrease the prefetch multiplier when:

- Host sequentiality is broken up due to use of a striped volume on the host side.
- I/O sizes close to that of the maximum prefetch value are used.
- Navisphere Analyzer shows that prefetches are not being used.

High and Low Watermarks and Flushing

The CLARiiON design has two global settings called *watermarks*—high and low—that work together to manage flushing. For most workloads, the defaults afford optimal behavior:

- FC series — High watermark of 60 percent and a low watermark of 40 percent.
- CX series — High watermark of 80 percent and a low watermark of 60 percent.

Increase the high watermark only if Navisphere Analyzer data indicates an absence of forced flushes during a typical period of high utilization. Decrease the high watermark if write bursts are causing enough forced flushes to impact host write workloads such that applications are affected. This reserves more cache pages to absorb bursts.

The low watermark should be 20 percent lower than the high watermark.

Write-Aside Size

Write-aside helps keep large I/O from taking up write cache mirroring bandwidth, and makes it possible for the system to exceed the write cache mirroring maximum bandwidth.

To exceed the write cache mirroring bandwidth, there must be sufficient drives to absorb the load, and either:

- The LUNs are RAID 0, RAID 1, or RAID 1/0.

Or, if a parity RAID (RAID 5 or RAID 3[5]) is used:

- I/O is equal to or a multiple of the LUN stripe size, *and*
- I/O is aligned to the stripe, *and*
- The LUN stripe size is 512 KB or less, *and*
- The stripe element size is 128 blocks or less.

These conditions for parity RAID are crucial and cannot be stressed enough. Getting I/O to align for effective write-aside can be difficult. If in doubt, use write cache. The tradeoff for doing write-aside is as follows:

- The data written this way is not available in cache for a subsequent read.
- The response times for writes are longer than for cached writes.

The write-aside size, as set in Navisphere, is the *largest I/O that will be write cached*. The default is 1023 blocks, so I/O of 512 KB and higher bypass cache.

For example, if the application executes writes of 256 KB and higher, and keeps 60 or more drives concurrently busy *doing writes*, setting the write-aside size to 511 blocks may increase performance—as all writes of 256 KB and higher bypass the cache. This effect increases as the drive count increases.

On the other hand, a CX600 can effectively cache I/Os up to 1 MB in size and keep up to 60 drives busy absorbing those writes. Increasing the write-aside size to 2048 blocks improves bandwidth and response times by allowing the cache to handle all writes up to and including 1 MB in size.

For CX-series users, it is suggested to change the write-aside size to 2048 blocks unless there is a clear need to use write-aside.

The Navisphere CLI `getlun` command displays the write-aside size for a LUN.

To change the write-aside size, use the Navisphere CLI `chglun` command with the `-w` parameter flag. In the following example, the `-l 22` flag indicates the action is on LUN 22, and the write-aside is being adjusted so that I/Os of up to 1 MB will be cached:

```
navicli –h <ip_address> chglun –l 22 –w 2048
```

Balancing Cache Usage between SPs

Lastly, ensure that the write cache usage is balanced *between SPs*. The amount of cache each SP is allocated is adjusted so that if more write I/O is coming through an SP, it will get more than half of the write cache. See Figure 10. This adjustment is done every 10 minutes.

Balance the storage system by ensuring that each SP owns an equal number of LUNs using the write cache.

[5] RAID 3 does not use write cache, so these conditions always apply to high bandwidth on RAID 3.

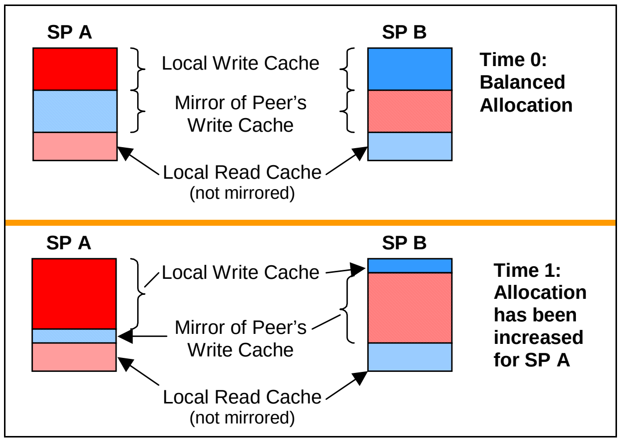
SP A
SP B
Local Write Cache
Mirror of Peer's Write Cache
Local Read Cache (not mirrored)
Time 0: Balanced Allocation
SP A
SP B
Local Write Cache
Mirror of Peer's Write Cache
Local Read Cache (not mirrored)
Time 1: Allocation has been increased for SP A

Figure 10. Write Cache Auto-Configuration

The Back End

This section addresses the assignment of drives to RAID groups and the distribution of LUNs.

LUN Distribution

For the purposes of this discussion:

- *Back-end bus* refers to the redundant pair of Fibre Channel loops (one from each SP) by which **all** CLARiiON storage systems access disk drives. (Some CLARiiON storage systems have dual back-end buses—a total of four fiber loops).
- A RAID group partitioned into multiple LUNs, or a LUN from such a RAID group, is referred to as a *partitioned RAID group* or *partitioned LUN*, respectively.
- A RAID group with only one LUN is called a *dedicated RAID group* and a *dedicated LUN*, respectively.

For efficient distribution of I/O on Fibre Channel drives, distribute LUNs across RAID groups. When doing distribution planning, take the capacity of the LUN into account. Calculate the total GB of high-use storage, and distribute the capacity appropriately among the RAID groups.

Additionally, balance load across storage processors. To do this, assign SP ownership: the default owner property for each LUN specifies the SP through which that LUN is normally accessed.

When partitioning ATA drive RAID groups, keep all LUNs from any RAID group owned by a single SP.

As a coda to the above note, to avoid ownership conflict, it is best to assign all LUNs from each ATA group to a single host. Otherwise a path-induced trespass on one host will cause the ownership of its LUNs to conflict with others on the same RAID group.

When planning for metaLUNs, note that all LUNs used for a metaLUN will be trespassed to the SP that owns the base LUN; their original default owner characteristic will be overwritten. Thus, when planning for metaLUNs, designating pools of SP A and SP B LUNs assists in keeping the balance of LUNs across SPs even.

Vault and Boot LUN Effects

In CX series systems, the first five drives in the base disk enclosure are used for several internal tasks.

Drives 0 through 4 are used for cache vault. The cache vault is only accessed when the system is disabling write cache (or enabling after a fault). Thus, there is no effect on host performance form the vault activities unless there is a fault.

The first four drives are also used as operating system boot and system configuration. Once the system has booted, there is very little activity from the FLARE operating system on these drives. Again, this will not affect host I/O.

Navisphere uses the first four drives for caching NDU data. Heavy host I/O during an NDU can cause the NDU to time out, so it is recommended that before an NDU commences, the host load be reduced to 100 IOPS per drive. This is covered in some detail in Appendix A of CLAR-PSP-084.

Also, very heavy host I/O on these four drives results in increased response times for Navisphere commands. Thus, for performance planning purposes, it is suggested to consider these drives as already having a LUN assigned to them. Distribute load accordingly.

Using LUN and RAID Group Numbering

This suggestion does not help performance but does assist in the administration of a well-designed system. Use RAID group numbering and LUN numbering to your advantage.

For example, number LUNs such that all LUNs owned by SP A are even numbered, and LUNs owned by SP B are odd numbered.

A scheme to extend this is to use predictable RAID group numbering, and extend the RAID group number into the LUN number. This will facilitate selection of LUNs for metaLUNs. The RAID group number embedded in the LUN number allows you to select LUNs from multiple RAID groups. See Table 2.

Table 2. Example of RAID Group and LUN Numbering

RAID Group	LUN	Default Owner
10	100	SP A
	101	SP B
20	200	SP A
	201	SP B
30	300	SP A
	301	SP B

For example, if selecting LUNs with which to extend FLARE LUN 101 into a metaLUN, choose LUNs 201 and 301. Why? All three LUNs belong to the same SP (and metaLUN components will all be trespassed to the same SP as the base LUN anyway). Also, now the I/O for the new metaLUN 101 will be distributed across three RAID groups.

Minimizing Disk Contention

As drive sizes continue to increase, partitioned RAID groups are more common, and it becomes more difficult to optimize disk behavior. The CLARiiON design is quite flexible and will deliver good performance, even with a significant amount of disk contention. However, for high-performance environments, the following guidelines apply.

Backup during Production

Environments that require sequential reads (online backups) concurrent with production will get very good results with RAID 1/0 groups, as the read load can be distributed across many spindles. RAID 5 can also deliver good read throughput while under moderate load, such as messaging applications. Such arrangements should be tested before deployment.

Snapshot Save Areas and BCV LUNs

It is not wise to place snapshot cache LUNs on the same drives as the source LUNS you will snap. Write operations will result in very high seek times and disappointing performance.

The same holds true for BCV LUNs: put them on disk groups separate from the LUNs they are cloning.

Stripes and the Stripe Element Size

The default stripe element size (128 blocks or 64 KB) is recommended, and should be used—except in unusual circumstances. Do not change this value unless instructed by Performance Engineering or an application-specific Best Practices white paper.

CLARiiON RAID 5 Stripe Optimizations

The requirements to achieve Modified RAID 3 (**MR3**) optimization for RAID 5 have been misunderstood for some time. Many EMC personnel believe that the CLARiiON RAID optimizations work only with a 4+1 or 8+1 stripe, which is *not true—MR3 works for any size RAID 5 group*.

When an I/O fills the RAID stripe, whether because it bypassed cache and was aligned on the stripe, or if sequential I/O is cached until it fills the stripe, if the following requirements are satisfied, MR3 will be used to write the data to the disks:

- The cache page size is 4 KB or, for stripes of 512 KB and larger, 8 KB.
- The stripe element size is 64 KB (128 blocks) or smaller, and a multiple of 8.

For example, with a 12+1 RAID 5 group and a 64 KB stripe element, the stripe size is 12*64 KB = 768 KB. For MR3, a cache page size of 8 KB or larger must be used. The 64 KB element is an even multiple of the cache page size (8 KB).

Having a disk group of 2+2, 4+1, or 8+1 *does* make it easier to align the stripe size to common host I/O sizes and still maintain aligned stripe element sizes.

Uncached Writes, Parity RAID, and MR3

The write cache imposes a maximum write bandwidth that the system can sustain. Bypassing write cache allows the system to achieve higher write loads, providing there are enough disks to deliver the performance. This is difficult when using parity RAID types (RAID 5 or RAID 3) because these RAID types depend on MR3 for high write performance.

While it is possible to achieve MR3 when the write cache is bypassed, it is more challenging.

Release 13 and later

In order to achieve MR3 when bypassing cache with Release 13 and later revisions of FLARE, you must meet these requirements in addition to the general requirements for MR3 listed earlier:

- The host request size must be a multiple of the stripe size.
- The host I/O must be aligned to the stripe.

Release 12 and earlier

In order to achieve MR3 when bypassing cache with older revisions of FLARE, in addition to the general requirements for MR3 listed earlier *and* the Release 13 requirements, the RAID group must have eight or fewer data drives (in other words, 8+1 or less).

Number of Drives per RAID Group

For bandwidth operations, it is more effective to maximize sequentiality on a small number of drives than to distribute a sequential load over many drives. Attempting to distribute high-bandwidth streams over too many drives results in:

- Less sequential access at the drives.
- Longer synchronization times as the processor waits for multiple devices to complete an I/O.

These effects are more pronounced when write-aside is in use: with cached I/O, the cache can insulate the host from increased synchronization times. When using write-aside, the I/O cannot complete until all drives complete the transfer.

For high bandwidth, very large disk groups (more than 10 drives) should typically be avoided because there is additional seek latency as all the disks align on the same stripe for a particular I/O. This is one reason why, under certain circumstances, a RAID 5 LUN performs as well in writes as a RAID 1/0 LUN: it has fewer spindles to synchronize when writing the entire stripe. For writing large I/O (512 KB or greater) with RAID 5, 8+1 drives is the maximum for most users.

Generally, large disk sets are more effective for random workloads than sequential workloads.

Large Spindle Counts

Distribution of data across many disks is effective for random-access workloads characterized by the following conditions:

- Many concurrent processes or threads
- Heavy asynchronous accesses

A large disk count allows concurrent requests to execute independently. For workloads that are random and bursty, striped metaLUNs are ideal. MetaLUNs that share RAID groups ideally have their peaks at different times. For example, if several RDBMS servers share RAID groups, activities that cause checkpoints should not be scheduled to overlap.

How Many Disks to Use in a Storage System

There are plateaus in performance where adding disks does not scale workload linearly. These are addressed in detail in the *Sizing the Storage Requirement* section on page 33, but the following are some rough guidelines for *strictly maximizing performance*. Refer to the appropriate performance white papers for the CX400 and CX600 for details on their metrics.

The drive counts presented in Table 3 are for *concurrently active* drives, under constant and moderate to heavy load.

Table 3. System High-Efficiency / High-Performance Drive Counts

For absolute best performance small I/O, random access, drives per system	
FC4700	60
CX700	200
CX600	160
CX500	120
CX400	60
CX300	60
CX200	30
For absolute best performance large I/O, sequential access, drives per system	
FC4700	40
CX700	80
CX600	40
CX500	40
CX400	20
CX200, CX300	20

Note: These considerations are for cases where the top priority is performance. As drives are added to the systems listed in Table 3, performance increases; however, the increase may not be linear.

Disk Types and Sizes

Currently, CLARiiON systems are available with, or have installed, the following drive types and sizes:

- 7,200 rpm Fibre Channel drives (181 GB)
- 10,000 rpm Fibre Channel drives (9, 18, 36, 73, 146 GB)
- 15,000 rpm Fibre Channel drives (18, 36, and soon also 73 GB)
- 5,400 rpm ATA drives (250 GB)

Effect of Drive Size

Although the drive manufacturers make much of increased data density, and thus higher transfer rates, on the larger capacity drives, the layers of protocol between the drive and the host insulate the drive's internal transfer rates from host access rates. In short, the effect is very small.

The size of the drive affects the *spindle count* for a given *capacity*, and that affects overall performance. More spindles per gigabyte equate to higher performance potential. Spindle for spindle, the drive rotational speed and technology are far more significant than the drive capacity.

Rotational Speed

For a given drive technology, rotational speed is the major differentiating factor. Higher rotational speed allows the drive to seek linearly along the tracks in less time. Furthermore, the higher-rpm drives include improvements in lateral seeks. Transfer rates are higher for higher-rpm drives, though the drive buffer and the Fibre Channel protocol limit the practical transfer rate of these drives.

Fibre Channel Drives

Fibre Channel drives are by definition enterprise-class devices. These drives feature on-disk firmware capable of queue reordering, buffering, and advanced seek optimizations. Many refinements including aerodynamic tuning of the heads, platters, and of the air currents in the drive allow them to perform at levels not achievable by desktop-drive standards. Rotational speed has a great affect on Fibre Channel drive performance, as these drives can leverage increased spindle speeds very effectively.

The rotational speed of the drive has a direct impact on random read performance, and a secondary impact on random writes. Random read performance is crucial, as storage processor cache hits are rare in open systems, and in most cases, the drive must be accessed. A faster rotational speed means less latency for random reads.

For random writes, which are absorbed by the SP write cache, the effect of rotational speed is seen in cache flushing. Faster drives can flush the cache faster than slower drives. A cache that flushes faster allows a higher rate of I/O to the storage system before watermark and forced flushing cause service times to increase.[6]

As a result, the 15K rpm drives offer about a 30 percent real-world increase in maximum random load on a 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. Reads, however, will benefit in any case.

Mixing Disk Types in a DAE

A mix of 10K and 15K rpm drives may be used within an enclosure. Keep drives the same within each group of five, and use a maximum of one speed change per enclosure.

ATA Drives

ATA drives are **not** recommended for random-access environments. The ATA specification was not designed for a heavily random multithreading environment.

In tests of raw speeds, with random I/O, the ATA drives have about one-third to one-fourth the ability to service I/O, with the greatest difference being with smaller I/O sizes and at higher thread counts.
See Table 4.

[6] Refer to the *Storage-System Cache* in *EMC CLARiiON Fibre Channel Fundamentals* for details on watermark and forced flushing.

Table 4. Random Access Performance of ATA Drives Relative to 10K rpm Fibre Channel Drives

Threads Per RAID Group	2 KB to 8 KB I/O Size	32 KB I/O Size
1	50%	50%
16	25%	35%

However, as mentioned in *RAID Levels and Performance*, in sequential operations, with large I/O sizes and few threads (single thread per disk group), the ATA drives perform much better, particularly with RAID 3 improvements in Release 13.

ATA Drives and RAID Levels

The ATA drives benefit greatly in bandwidth applications with the Release 13 RAID 3 enhancements.

Recent testing has proved that RAID 1/0 performance is much better for BCV (clone) implementations than RAID 5, as clone usage puts stress on the write cache.

The best choice as RAID type for ATA drives is:

- Bandwidth applications (backup to disk, rich media): RAID 3 (write cache on)
- Replication (Snapshot Save Area, BCV): RAID 1/0

BCVs that are normally fractured, and then periodically synchronized, should use RAID 3, as synchronizing is a high-bandwidth operation.

RAID Group Partitioning and ATA Drives

When partitioning a RAID group with ATA drives, assign all LUNs from that RAID group to the same SP. This improves throughput at the drive level. (This is *not* a requirement for Fibre Channel drives.)

ATA Drives as Mirror Targets and BCVs

- One of the primary recommendations for ATA drives is for use as SnapView targets and BCVs. What is the performance impact in this implementation?

In a system that is not being stressed (i.e., write cache not hitting forced flushes), the use of ATA drives compared to FC drives has no significant effect on performance.

In a system that is already experiencing forced flushes, a synchronization of a BCV or mirror, or the establishment of a BCV or mirror implemented on ATA drives could cause the write cache to fill. This would negatively affect all I/O on the system.

Of course, reads from a fractured BCV or mirror will be at ATA speeds.

Considerations for Reliability and Redundancy

A reliable and redundant storage network starts with SAN design, which is beyond the scope of this paper. However, some aspects of SAN design are inherently storage-system subjects.

Highly Available Attach

Highly available attach methodologies are definitely a best practice for implementing CLARiiON storage. The *EMC CLARiiON Open Systems Configuration Guide* outlines the attach methodologies and equipment that are supported.

RAID Level Considerations

Most CLARiiON storage is implemented with RAID 1/0 or RAID 5 groups, as the redundant striped RAID types deliver the best performance and redundancy. RAID 3 is as redundant as RAID 5 but the implementation is restricted on CLARiiON arrays, so there is little to address regarding RAID 3.

RAID 5

RAID 5 is best implemented in five- to nine-disk RAID groups. The main drawback to large groups is the amount of data affected during a rebuild. The time to complete a rebuild is also longer with a larger group, though binding large RAID 5 groups across two back-end buses can minimize the effect. See Table 5 for details on rebuild times. 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.

For systems where slowdowns due to disk failure could be critical, or where data integrity is critical, use a modest number of spindles per RAID group. Better yet, use RAID 1/0.

RAID 1/0

Use RAID 1/0 when availability and redundancy are paramount. By nature, mirrored RAID is more redundant than parity schemes. Furthermore, a RAID 1/0 group needs only two DAEs—one from each back-end bus—in order to afford the highest possible level of data availability. Refer to the *Binding across DAEs* section on page 31.

The advantages of RAID 1/0 to RAID 5 when under rebuild are illustrated in Table 5.

Table 5. RAID Types and Relative Performance in Failure Scenarios

RAID Type	Rebuild IOPS Loss	Rebuild Time	Impact of Second Failure during Rebuild
RAID 5	50%	15% to 50% slower than RAID 1/0	Loss of data
RAID 1/0	20% to 25%	15% to 50% faster than RAID 5	Loss of data 14% of time in an eight-disk group (1/[*n*-1])
RAID 1	20% to 25%	15% to 50% faster than RAID 5	Loss of data

Binding RAID Groups across Buses and DAEs

Engineers with experience with older SCSI-based systems expect to bind parity RAID groups with each disk in a separate enclosure. This was done when each enclosure was served by a separate SCSI bus. Considerations for availability due to SCSI failure semantics no longer apply. However, there are several things to consider when binding disks.

Binding across DAEs

Few subjects cause as much concern and confusion in the field as the binding of disks across DAEs. Is there a performance advantage? Is there a redundancy advantage? In both cases, it depends on the RAID configuration, and in all cases the differences are slight.

Parity Groups (RAID 3, RAID 5)

Binding parity RAID groups such that each drive is in a separate DAE does not impact performance. However, there is a small increase in data availability in this approach. Using a parity RAID type with the drives striped vertically increases availability to over 99.999 percent. However, this is very unwieldy; if very high availability is required, use RAID 1/0. Refer to *Binding Across Back-End Buses* on page 31.

RAID 1/0 Groups

There is absolutely no advantage in binding a RAID 1/0 group in *more than* two DAEs, but it certainly is not harmful in any way.

Binding across Back-End Buses

All current CLARiiON systems except the CX200 and CX300 have dual redundant back-end buses with which they attach to the DAEs. A disk group can be made up of drives from one or both buses. The standard racking alternates the buses across adjacent DAEs, with the DPE or first DAE being bus 0, the next DAE bus 1, and so on.

Parity Groups (RAID 3, RAID 5)

Parity groups of 10 drives or more benefit from binding across both buses, as this helps reduce rebuild times. For example, bind a 10-drive RAID 5 with five drives in one DAE, and another five drives in the next DAE above it.

Mirrored Groups (RAID 1, RAID 1/0)

Binding mirrored RAID groups across two buses increases availability to over 99.999 percent and keeps rebuild times lower. This technique ensures availability of data in two (rare) cases of double failure: an entire DAE[7] or redundant back-end bus (dual-cable failure). Bind the drives so that the primary drives for each mirror group are on the first back-end bus, and the secondary (mirror) drives are on the second back-end bus. Binding across buses also has a minimal but positive impact on performance.

To bind across buses, when creating the RAID group (or defining a dedicated LUN in the bind command), use Navisphere CLI. When designating the disks, Navisphere CLI uses the disk ordering given in the `createrg` or `bind` command to create Primary0, Mirror0, Primary1, Mirror1, and so on, in that order. Disks are designated in `Bus_Enclosure_Disk` notation. Here is an example of binding the first two drives from enclosure one of each bus:

```
Navicli –h <ip address> createrg 55  0_1_0  1_1_0  0_1_1  1_1_1
```

Binding with DPE Drives

In a total powerfail scenario, the SPS (standby power supply) supplies battery-backed power to the SPs and vault disks. This allows the storage system to save the contents of the write cache to disk.

However, the power to the non-vault disk storage-system enclosures (DAEs) is not maintained. When the storage system reboots, LUNs that had I/O outstanding are checked, using the background verify process, to verify that no writes in progress resulted in partial completions. The background verify is a fairly low-intensity process.

However, a LUN bound with some drives in the vault enclosure (DPE or first DAE, depending on the model) and with some drives outside of the vault enclosure *may* require a rebuild, which is a more disk-intensive process. This affects performance to some degree on reboot.

To avoid a rebuild on boot, follow these steps:

- Do not split RAID 1 groups across the vault enclosure and another DAE.
- For Parity RAID (RAID 5, RAID 3), make sure at least two drives are o*utside* the vault enclosure.
- For RAID 1/0, make sure at least one mirror (both the primary and secondary drive in a pair) is outside the vault enclosure.

For RAID 1/0, you can use NaviCLI's ordering when using `createrg`, as explained earlier to ensure at least one pair is outside the vault enclosure. Example:

```
Navicli –h <ip address> createrg  45  0_1_0  1_1_0  0_0_1  1_0_1  0 0 2  1 0 2
```

Note that the pair 0_1_0 and 1_1_0 are outside the vault enclosure.

Or simply ensure more than one-half the drives in a RAID 1/0 group are outside the vault enclosure.

Consistency of Data Replication

The use of SnapView, MirrorView, and SAN Copy are out of the scope of this paper, but the concept of consistency is very close to that of redundancy, and thus warrants attention here. (Refer to the EMC white papers covering these other products.)

[7] The active components of a DAE are redundant (power supply, cooling, LCC). The passive components are redundant as well, and are not subject to mechanical or thermal stress during normal operations.

When implementing mirroring over distance or replication of any kind (including backups) of **multi-LUN related data**, the engineer *must* take consistency into account. Multi-LUN related data is data that spans more than one LUN and is interrelated. Examples are:

- Components of a relational database: data tables, logs, TEMP and RBS
- Volumes in host volume sets (VERITAS, PowerVolume)
- Components of a messaging database
- Shared file systems

Multi-LUN consistency depends on certain behaviors in the storage system. For example, writes to a database that spans LUNs must remain consistent across those LUNs. If a transaction is marked committed in the log, the data must be in the tables. The two cases in which the storage design must address this requirement are hot splits and mirroring.

In a hot split, BCVs are fractured, or snapshots are taken, of all related LUNs while host access is concurrent. For example, if you fracture a BCV and:

- Host I/O is ongoing to a messaging application, or
- An RDBMS is running, or
- A file system has not been *synced* (buffers flushed to disk).

If the data spans multiple LUNs, the split must take effect across all LUNs at the same time.

(In a *warm split*, either host I/O is quiesced and the data is synchronized to disk or, in the case of some RDBMS systems, the database is held in *hot backup mode*. In this case, the BCVs and snapshots must be taken while the data is held in this mode, but they need not be taken simultaneously with each other.)

Mirroring of data is the second case. Consistency requires that if there is an interruption in the mirroring—to a BCV or remote mirror image—of one LUN, that all LUNs that make up the related data set must stop their mirrors. Otherwise, the mirror image becomes inconsistent. Since MirrorView maintains connectivity on an SP basis, multiple-LUN databases should be mirrored from a set of LUNs that resides on the same SP. In the case of a volume group, this is an argument for metaLUNs in favor of host volume groups—there is only one LUN to mirror for a metaLUN.

Currently, there is no way to automate the synchronizing of MirrorView or SnapView operations across multiple LUNs. The capability, referred to as *consistency groups*, is one that CLARiiON engineering will address in 2004.

Sizing the Storage Requirement

Storage sizing consists of calculating the right number of drives for capacity, and calculating the correct number of drives and the right storage system for performance.

Capacity Planning

Vault Drives

The first five drives in a CX series storage system contain the vault. Vault drives can be used just as any other drives on the system. However, vault drives have less useable capacity. When bound with other, nonvault drives, all drives in the group are foreshortened to match the vault drive capacity. It is thus a best practice to bind the vault drives together as a group (or groups).

Actual Drive Capacity

The capacity of the drives is much less than the rated capacity. This is mainly because manufacturers consider a gigabyte to be 1,000,000,000 bytes (base 10 gigabyte). A computer uses base 2 (binary) and a binary gigabyte is 1,073,741,824 bytes. Many engineers are surprised to find a 36 GB drive holds only 33 GB.

Also, CLARiiON arrays use eight additional bytes per sector for storing redundancy information. This reduces the useable capacity by a small margin.

Performance Planning

Performance planning or forecasting is a science that takes considerable knowledge. The steps presented here are intended for rough estimation only.

Rule-of-Thumb Approach

To begin a performance estimation, a rule of thumb is used for IOPS per disk drive and MB/s per disk drive (see Table 6). This is a conservative and intentionally simplistic measure. It should be noted this is only the beginning of an accurate performance estimate, and estimates based on the rule of thumb are for quickly sizing an account. More accurate methods are available to presales escalation personnel.

The metrics in Table 6 assume:

- IOPS figures assume 2 KB to 8 KB random requests
- MB/s (bandwidth) figures assume 128 KB and larger sequential requests

The CX500 and CX700 have higher rule-of-thumb measures for IOPS due to an improvement in their write cache flushing algorithm.

Table 6. Data Access Rate Guidelines for Fibre Channel Disk Drives

Drive Speed	**IOPS (8 KB Random Mix)**		**Bandwidth (256 KB Sequential Mix)**
	Standard	**CX500/700**	
5,400 rpm ATA	N/A	N/A	5 MB/s
10,000 rpm FC	100 IOPS	120 IOPS	10 MB/s
15,000 rpm FC	150 IOPS	180 IOPS	12 MB/s

The approach for a quick estimate is:

1. Determine host I/O or bandwidth load.
2. Calculate disk I/O or bandwidth load.
3. Calculate number of disk drives required for disk I/O or bandwidth load.
4. Calculate number and type of storage systems.

Determining Host Load

This is often the most difficult part of the estimation. Some sort of estimate must be made. The estimate must include not only the total IOPS, but also what percentage of the load is reads and what percentage is writes. Additionally, the predominant I/O size must be determined.

Determining Disk Load

Note that the IOPS values in Table 6 are *disk IOPS*. To determine the number of disk IOPS implied by a host I/O load, adjust as follows for parity or mirroring operations:

Parity RAID: *Disk IOPS = Read IOPS + 4*Write IOPS*

Mirrored RAID: *Disk IOPS = Read IOPS + 2*Write IOPS*

For bandwidth figures, use the MB/s figure estimated above.

Calculate Disk Drives Required

Divide the total IOPS or bandwidth by the per-disk IOPS or bandwidth figure in Table 6. This is the approximate number of drives you need to service the proposed I/O load. If performing random I/O with a predominant I/O size above 8 KB, increase the disk count by 10 percent to 20 percent.

Add one hot spare drive per 30 drives to the drive count.

Calculate Number and Type of Storage Systems

Once the number of drives is estimated, they must be matched to a storage system or set of systems that supplies performance, capacity, and value.

High Performance Requirement

Refer to Table 3 to select the storage system whose high-efficiency/high-performance drive count—the sweet spot—best fits the requirement.

Divide the number of drives by the high-efficiency/high-performance drive count to determine the number of storage systems necessary to service the required disk drives effectively.

Low Performance Requirement

For low-performance uses, use up to the maximum drive count for the chassis. Divide the number of disk drives by the chassis maximum.

Rectifying Performance and Capacity Needs

Select a drive size that, when matched to the RAID type and group size and spindle count needed for performance, matches the required capacity.

Sizing Example

The following example uses the procedure outlined earlier.

Step 1: Determine the Required I/O Load

Client X has a requirement for 20 TB of storage for an object database. The application is expected to execute many random 16 KB I/O. The profile is:

- 70 percent random reads
- 30 percent random writes
- Predominant I/O size is 16 KB
- Total IOPS is 14,000

The client is price-sensitive, so nine-disk RAID 5 is desired.

Step 2: Calculate the Disk Load

We plan to use a parity RAID, so the disk load is:

0.7 * 14,000 + 4 * 0.3*14,000 = 26,600 disk IOPS

Step 3: Calculate Number of Drives Required

For 26,600 disk IOPS, we will need 26,600 / 100 = 266 disk drives. The I/O size is a bit larger than our rule-of-thumb size, so add 10 percent—another 25 drives for a total of 291 drives. Nine hot spares are needed, for a total of 300 drives.

Step 4: Calculate Number and Type of Storage Systems

Certainly a drive count of this magnitude calls for a pair of CX600s (we would need five CX400 systems for this many drives) as a single CX600 can support 160 drives in random-access, small block operations—the sweet spot for random I/O from the *CX600 Performance Paper*, on Powerlink.

Rectifying with Capacity

290 of the 73 GB drives in 8+1 RAID 5 format would yield 16.5 TB—some 146 GB drives can be substituted to reach the 20 TB requested.

Summary

Performance, reliability, redundancy, and capacity are the primary considerations for choosing storage.

With a CLARiiON Fibre Channel storage system, getting very good performance—what 80 percent of our customers need—is fairly straightforward. To squeeze the last bit of performance from the storage system, the operator must be aware of host-based effects and the geometry of the RAID configuration. The applications analysis and host memory configuration *must* be done before tuning the storage system.

Reliability is built into the storage system—with greater than 99.99 percent availability. Redundancy is easy to achieve, but more costly than non-highly available configurations. Much of the redundancy of the CLARiiON storage system is built in—dual SPs, hot-swappable disks, redundant RAID, and hot spare disks. The choices made in the field are in the attach methodology and RAID types chosen. For the cost-conscious client, the costs of *not* building redundancy into the attach configuration must be explained. RAID 1/0 is the gold standard for redundancy.