UNIX I/O Performance Measurement Methodologies Applied To Old And New Storage Technologies

Mark Cohen Austrowiek – Demand Technology Software
Pierluigi Grassi – EMC Computer Systems

ABSTRACT

In this document we'll provide a general overview of UNIX I/O performance and their operating system tools that measure internal server disks and how these tools can be applied to modern external storage arrays and eventually a SAN environment. In particular we'll describe the current storage architecture by means of integrating native system tools and storage vendor software in order to obtain information as performance measurements, physical disk mapping and more. In addition we'll provide some considerations when planning and configuring the I/O systems in order to improve performance.

INDEX

1 I/O PERFORMANCE OVERVIEW 3

2 HARDWARE CONFIGURATION 4

3 VOLUME GROUP AND FILESYSTEM CONCEPTS 6

3.1 PHYSICAL VOLUMES 6
3.2 VOLUME GROUPS 6
3.3 LOGICAL VOLUMES 6
3.4 LIMITATIONS FOR LOGICAL STORAGE MANAGEMENT 6
3.5 FILE SYSTEMS OVERVIEW 7
3.6 PLACEMENT AND REORGANIZING A LOGICAL VOLUME, VOLUME GROUP OR FILEYSTEM 7

4 STORAGE DISK ARRAY 9

5 NATIVE UNIX UTILITIES TO ANALYSE THE I/O SYSTEM 12

5.1 FILEMON STATISTICS 16

6 ROTS - RULES OF THUMB 17

6.1 CHANNEL BANDWITH AND BLOCKSIZE 17
6.2 ACCESS DENSITY 17

7 STORAGE VENDOR UTILITIES 18

7.1 CONTROL CENTER OPEN EDITION (CCOE) 19
7.2 CC SYMMETRIX MANAGER 19
7.3 CC WORKLOAD ANALYZER 19
7.4 CC OPTIMIZER 19

8 CONCLUSIONS 20

9 APPENDIX A – EMC UTILITIES' ARCHITECTURE AND GRAPHS 21

9.1 CONTROL CENTER OPEN EDITION ARCHITECTURE 21
9.2 SYMMETRIX MANAGER UTILIZATION EXAMPLE 21
9.3 WORKLOAD ANALYZER 26

1 I/O Performance Overview

Improving system I/O performance has always been a difficult task for performance analysts. The steps needed to analyse I/O performance are:

- Verify that other resources as memory, CPU or network activities are not the real cause of the I/O performance degradation.
- Controlling service levels as transactions per second, disk busy and disk response times (not provided by most UNIX platforms) help in locating I/O performance problems.
- Modifying operating system parameters to improve read and write ahead activity, page stealing policy, I/O pacing and logical volume implementation as striping, partition sizing , mirroring and data positioning have major effects on I/O performance.
- Understanding how the application is accessing data (sequential or random access, reads vs writes) and the block size is essential in order to choose the correct raid..

In addition, nowadays the logical and physical path concepts have been extended due to the fact that many systems access the same storage controllers, sharing the same physical path through switches (as in the mainframes environment where different systems share the same physical paths from ESCON DIRECTORS to the Storage Control Unit).

Due to these new technologies I/O performance in the Open environment has become more complex to understand and questions like where it is done, how is the I/O performing, where is the I/O bottleneck and what kind of storage configuration would be best for my I/O are becoming more difficult to answer.

In a simple system where we find a simple bunch of disks there is a direct correspondence between the physical disk shown by the native UNIX utilities and the physical disk installed on the system. We can easily understand the throughput and the amount of time the disk was busy writing and reading data.

When using Storage Disk Array the correspondence between the logical entries found in the operating system and the internal physical disks are hidden. A logical disk can be associated with a different amounts of space and located on multiple physical drive locations in the storage control unit . This can be done on behalf of the storage administrator through logical volume manager (logical volumes), but more simply can be achieved configuring the volumes using the microcode installed in the storage controller (metavolumes) with less operating system overhead.

To get the real physical disk throughput activity and physical disk busy we would have to map all the logical entries to the physical disks and sum the throughput and busy for the physical disks associated with each logical entry for each system. This is quite a cumbersome procedure for a performance analyst, unless he/she has an integrated software interface with the storage physical configuration.
Using a software interface, like ECC from EMC or STORWATCH specialist from IBM, allows you to map the logical volumes, as they are presented to the operating system (for example an AIX hdisk0), to the physical disks they reside on. Having this information, makes it much easier to understand the locations of bottlenecks and how the I/O workload is balanced for the backend. Depending on the storage architecture the software interface may vary , but in any case the objective is to offer the user a graphical view of the internal storage array.

An EMC component also has the ability to discover and display the relationship between databases, filesystem, hosts and storage, so from a console management station you can determine that your Oracle tablespace is located on a particular filesystem, host logical volume, and array.
In this document we tried to provide a brief overview of the hardware configuration components and software available to analyse these components. Understanding the logical to physical disk and path mapping, the cache architecture, the raid level serves as the basis in understanding the system I/O potential.

2 Hardware Configuration

Acquiring a good knowledge of the characteristics of all the hardware components in the I/O physical path is very helpful in order to understand the theoretical I/O throughput capacity. The real throughput also depends on the operating system options, system and application software and storage disk array microcode.

Looking at the CPU, a 486 system has the ability to produce 45 MB/sec throughput while a Pentium based system can provide 132 MB/sec throughput. A PCI bus (64 bit peripheral component interface 33MHZ/66MHZ) can range from 237 to 532 MB/sec. IBM states that one mainframe MIPS is equivalent to a 60 pentium mhz bus.

The positioning of the bus slots is also a very important factor when installing adapters. There are specifications in the hardware configuration manual which recommend what adapters should be positioned in each slot. Adapters have different weights (low, middle, high) and can influence the adapters performance if they do not respect the installation recommendations

Channel adapters or HBA (host bus adaprter) are another important element in the I/O chain. There are different types of them and historically channel bandwidth has always been improved over the years: examples are NarrowSCSI 10 MB/s, FastSCSI (20MB/s), UltraSCSI (40MB/s), UltraSCSI 2 (80 MB/s), FC – fiber channel (100/200 MB/s). Throughput is strongly dependent on I/O blocksize and protocol overhead, so, for example, we could achieve the same performance with UltraSCSI and fiber channel when the block size is less than 2 K
Whether the performance is measured by response time, I/O per sec., MB per sec., there is no one number categorizing performance; it has to be correlated with the I/O size.
A significant improvement in performance has been achieved in the last few years through the introduction of I/O path balancing and path failover in the UNIX and Windows environment , a function available to Mainframe since the late 80's.

Historically the UNIX and Windows operating systems have one channel path to a specific volume, so even if different HBA's are mounted on the server each of them will access different disks. It's very common that the workload is not balanced across them and it's possible to loose access to a group of disks in case of an HBA failure, unless there is a cluster implemented. (some UNIX OS can partially manage the HBA failure).

With the introduction of a software layer between the kernel and the application layer (something like the mainframe channel subsystem) it's now possible to have a server with multiple HBAs accessing the same group of hdisks inside the Storage Disk Array. In case of a path failure, I/Os are redirected to the surviving paths.
I/O balancing can be done using different algorithms:

- Round Robin (rr)
  Paths assigned in rotation regardless of other factors.
- Least I/Os (li)
  Select the path with the least number of pending I/Os regardless of I/O size.
- Least Blocks (lb)
  Select the path with the least number of blocks pending regardless of the number of blocks per I/O.
- Request (re)
  Failover is in effect, but load balancing is not.
- No redirect (nr)
  Neither load balancing nor failover is in effect..

The EMC Powerpath can use all the above algorithms. It defaults to a Symmetrix Optimization algorithm which permits I/O request balanced across paths based on composition of channel speeds, number and size of read requests, write requests, sequential activity, and tunable priority value. Symmetrix Optimization (default) is the recommended algorithm as tests have shown that this option always out performs other options.

Disk drives are another key element when evaluating performance. With the introduction of **Storage Disk Arrays**, the architecture (microcode features, processors power, cache dimension and management algorithm, raid level protection, etc…) of the SDA has taken predominance over the disk characteristics itself. We will discuss Storage Disk Array later in the document and we' ll examine the backend I/O activity and how it can be impacted .from the disks characteristics.

The major components of disk performance are the following:

- Seek Time - required to move the disk drives heads from one cylinder to another. This obviously depends on how far the heads have to travel. Not as obviously , there is no linear relationship with the number of cylinders you need to cross over as the heads need to accelerate, decelerate and then stabilize in their new position. To reduce confusion, disks manufacturers typically specify a minimum seek (one cylinder to the next), an average seek time (the average time required for the heads to move from one track to any other track) and a maximum seek time.
- Latency - once the disk drives has moved the heads in to place it has to wait until the data you want has moved under the head. This requires at most one full rotation of the disk itself - on the average, the disk will have to spin one half turn. Therefore the disk rotational speed also determines the disk latency, or the amount of time it takes to get ready for a data transfer.
- Transfer rate - a disk's raw transfer rate is the speed at which it moves data. A program never receives useful data this fast; the raw transfer just measures the speeds at which bits come from the disk drive; it doesn't account for formatting data that your program never sees, wasted space on the disk, and other factors.

The following table contains some standard disk characteristics that are usually provided by different disk manufacturers. In our opinion the most important factors are spindle speed, external transfer rate and latency time.

	18GB LP 3.5" (Cheeta)	**36GB - 3.5" (Cheetah)**	**50GB - 3.5" (Cuda)**	**73 GB - 3.5" (Cheetah)**	**181 GB - 3.5" (Cuda)**
Interface	Ultra SCSI	Ultra SCSI	Ultra SCSI	Fast Wide SCSI-2	Fast Wide SCSI-2
Spindle Speed (rpm)	10.000	10.000	7.200	10.000	7.200
External transfer rate (MB/sec)	40	40	40	33	33
Actuator-level buffer	4MB	4MB	4MB	16 MB	16 MB
Minimum seek (msec)	0.6/0.9 (read/write)	0.8/1.1 (read/write)	0.9/1.2 (read/write)	1.1/1.4 (read/write)	1.1/1.4 (read/write)
Maximum seek (msec)	12-13 (read/write)	14.5/15.7 (read/write)	16/17 (read/write)	16/18 (read/write)	16/18 (read/write)
Average seek (msec)	5.7/6.5 (read/write)	6.15/6.85 (read/write)	7.6/8.4 (read/write)	7.9/8.7 (read/write)	7.9/8.7 (read/write)
Latency (msec)	2,99	2,99	4,17	4,17	4,17

Before we go further in depth, it's better to summarize some additional concepts from a server perspective.

3 Volume group and filesystem concepts

3.1 Physical Volumes

A disk must be designated as a physical volume and be put into an available state before it can be assigned to a volume group. A physical volume has certain configuration and identification information written on it. This information includes a physical volume identifier that is unique to the system. When a disk becomes a physical volume, it is divided into 512-byte *physical blocks*. You designate a disk as a physical volume with the **mkdev** or **chdev** commands or by using the System Management Interface Tool (SMIT) to add a physical volume.
The first time you start up the system after connecting a new disk, the operating system detects the disk and examines it to see if it already has a unique physical volume identifier in its boot record. If it does, the disk is designated as a physical volume and a physical volume name (typically, **hdisk***x* where *x* is a unique number on the system) is permanently associated with that disk until you undefine it.

3.2 Volume Groups

The physical volume must now become part of a *volume group*. A volume group is a collection of 1 to 32 physical volumes of varying sizes and types. A physical volume may belong to only one volume group per system; there can be up to 255 volume groups per system.

3.3 Logical Volumes

After you create a volume group, you can create logical volumes within that volume group. A *logical volume,* although it may reside on noncontiguous physical partitions or even on more than one physical volume, appears to users and applications as a single, contiguous, extensible disk volume. You can create additional logical volumes with the **mklv** command. This command allows you to specify the name of the logical volume and define its characteristics, including the number and location of logical partitions to allocate for it. After you create a logical volume, you can change its name and characteristics with the **chlv** command, and you can increase the number of logical partitions allocated to it with the **extendlv** command. The default maximum size for a logical volume at creation is 128 logical partitions, unless specified to be larger. The **chlv** command is used to relax this limitation.

3.4 Limitations for Logical Storage Management

The following table shows the limitations for logical storage management. Although the default maximum number of physical volumes per volume group is 32 (128 in case of big volume group), you can set the maximum for user-defined volume groups when you use the mkvg command. For the rootvg, however, this variable is automatically set to the maximum by the system during the installation.

```
        MAXPVS: 32 (128 big volume group)
        MAXLVS: 255 (512 big volume group)
```

Limitations for Logical Storage Management

Volume group	255 per system
Physical volume	(MAXPVS / volume group factor) per volume group
Physical partition	(1016 x volume group factor) per physical volume up to 1024MB each in size.
Logical volume	MAXLVS per volume group
Logical partition	(MAXPVS * 1016) per logical volume

If you previously created a volume group before the 1016 physical partitions per physical volume restriction was enforced, stale partitions in the volume group are not correctly tracked unless you convert the volume group to a supported state. You can convert the volume group with the **chvg -t** command. A suitable factor value is chosen by default to accommodate the largest disk in the volume group.
For example, if you created a volume group with a 9GB disk and 4MB partition size, this volume group will have approximately 2250 partitions. Using a conversion factor of 3 (1016 * 3 = 3048) allows all 2250 partitions to be tracked properly. Converting a volume group with a higher factor enables inclusion of a

larger disk of partitions up to the 1016* factor. You can also specify a higher factor when you create the volume group in order to accommodate a larger disk with a small partition size.
These operations reduce the total number of disks that you can add to a volume group. The new maximum number of disks you can add would be a 32/factor. For example, a factor of 2 decreases the maximum number of disks in the volume group to 16 (32/2).
The Logical Volume Manager (LVM) consists of the logical volume device driver (LVDD) and the LVM subroutine interface library. The *logical volume device driver* (LVDD) is a pseudo-device driver that manages and processes all I/O. It translates logical addresses into physical addresses and sends I/O requests to specific device drivers. The *LVM subroutine interface library* contains routines that are used by the system management commands to perform system management tasks for the logical and physical volumes of a system. The programming interface for the library is available to anyone who wishes to expand the function of the system management commands for logical volumes.

3.5 File Systems Overview

A *file system* is a hierarchical structure (file tree) of files and directories. This type of structure resembles an inverted tree with the roots at the top and branches at the bottom. This file tree uses directories to organize data and programs into groups, allowing the management of several directories and files at one time.
Some tasks are performed more efficiently on a file system level than on each directory within the file system. For example, you can back up, move, or secure an entire file system.
A file system resides on a single logical volume. The **mkfs** (make file system) command or the System Management Interface Tool (**smit** command in AIX env.) creates a file system on a logical volume. Every file and directory belongs to a file system within a logical volume.

3.6 Placement and reorganizing a logical volume, volume group or fileystem

```
# smit reorgvg
```

The **reorgvg** command reorganizes the placement of allocated physical partitions within the *VolumeGroup*, according to the allocation characteristics of each logical volume. Use the *LogicalVolume* parameter to reorganize specific logical volumes; highest priority is given to the first logical volume name in the *LogicalVolume* parameter list and lowest priority is given to the last logical volume in the parameter list. The volume group must be varied on and must have free partitions before you can use the **reorgvg** command.

If the workload shows a significant degree of I/O activity, you can investigate the physical placement of the files on the disk to determine if reorganization at some level would yield an improvement. To see the placement of the partitions of logical volume `hd11` within physical volume `hdisk0`, use the following unix commands.

```
$ lslv -p hdisk0 hd11

hdisk0:hd11:/home/op
USED  USED  USED  USED  USED  USED  USED  USED  USED  USED     1-10
USED  USED  USED  USED  USED  USED  USED                      11-17

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED    18-27
USED  USED  USED  USED  USED  USED  USED                      28-34

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED    35-44
USED  USED  USED  USED  USED  USED                            45-50

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED    51-60
0052  0053  0054  0055  0056  0057  0058                      61-67

0059  0060  0061  0062  0063  0064  0065  0066  0067  0068    68-77
0069  0070  0071  0072  0073  0074  0075                      78-84
```

The word USED means that the physical partition is in use by a logical volume other than hd11. The numbers indicate the logical partition of hd11 that is assigned to that physical partition.
We look for the rest of hd11 on hdisk1 with:

```
$ lslv -p hdisk1 hd11

hdisk1:hd11:/home/op
0035  0036  0037  0038  0039  0040  0041  0042  0043  0044    1-10
0045  0046  0047  0048  0049  0050  0051                     11-17

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED   18-27
USED  USED  USED  USED  USED  USED  USED                     28-34

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED   35-44
USED  USED  USED  USED  USED  USED                           45-50

0001  0002  0003  0004  0005  0006  0007  0008  0009  0010   51-60
0011  0012  0013  0014  0015  0016  0017                     61-67

0018  0019  0020  0021  0022  0023  0024  0025  0026  0027   68-77
0028  0029  0030  0031  0032  0033  0034                     78-84
```

We see that logical volume hd11 is fragmented within physical volume hdisk1, with its first logical partitions in the inner-middle and inner regions of hdisk1, while logical partitions 35-51 are in the outer region. A workload that accessed hd11 randomly would experience unnecessary I/O wait time as the disk's accessor moved back and forth between the parts of hd11. These reports also show us that there are no free physical partitions in either hdisk0 or hdisk1.

```
$ fileplace -pv big1
```

Displays the placement of file blocks within logical or physical volumes.

The resulting report is:

```
File: big1  Size: 3554273 bytes  Vol: /dev/hd10 (4096 byte blks)
Inode: 19  Mode: -rwxr-xr-x  Owner: frankw  Group: system

Physical blocks (mirror copy 1)                     Logical blocks
-------------------------------                     --------------
01584-01591  hdisk0       8 blks,    32 KB,   0.9%    01040-01047
01624-01671  hdisk0      48 blks,   192 KB,   5.5%    01080-01127
01728-02539  hdisk0     812 blks,  3248 KB,  93.5%    01184-01995

  868 blocks over space of 956:  space efficiency = 90.8%
  3 fragments out of 868 possible:  sequentiality = 99.8%
```

This shows that there is very little fragmentation within the file, and those are small gaps. We can therefore infer that the disk arrangement of big1 is not affecting its sequential read time significantly. Further, given that a (recently created) 3.5MB file encounters this little fragmentation, it appears that the file system in general has not become particularly fragmented.

4 Storage Disk Array

A storage disk array is a unit which includes a set of disk drives, usually a cache and channel interface connected to the host and essentially microcode. Beyond this definition there are many different architectures, so it is important to evaluate the main characteristics and understand how performance can be impacted from them.

Microcode is the core of every disk array. It needs to be well tested, flexible, full of functionalities and has to be able to manage all the resources inside the array (channel interface or directors, cache, disk adapters and disks itself). Of course in order to achieve this goal it must rely on very powerful.
Having large cache inside the SDA is preferred but it's not a performance warranty if the microcode is not able to manage concurrent accesses, prefetching or destaging data according to I/O patterns.

Cache is the performance element of the SDA and has the goal to reduce I/O response time. Data are transferred to the channel at electronic speed instead of disk speed. Based on performance numbers available, a read I/O from an uncached disk would take approximately 12 msec., the same read served from cache would take almost 0,5 msec. It's evident the importance of resolving I/O as much as possible in cache, so the imperative should be "**cache hits are good....maximize cache hits" , but how can we do that?**

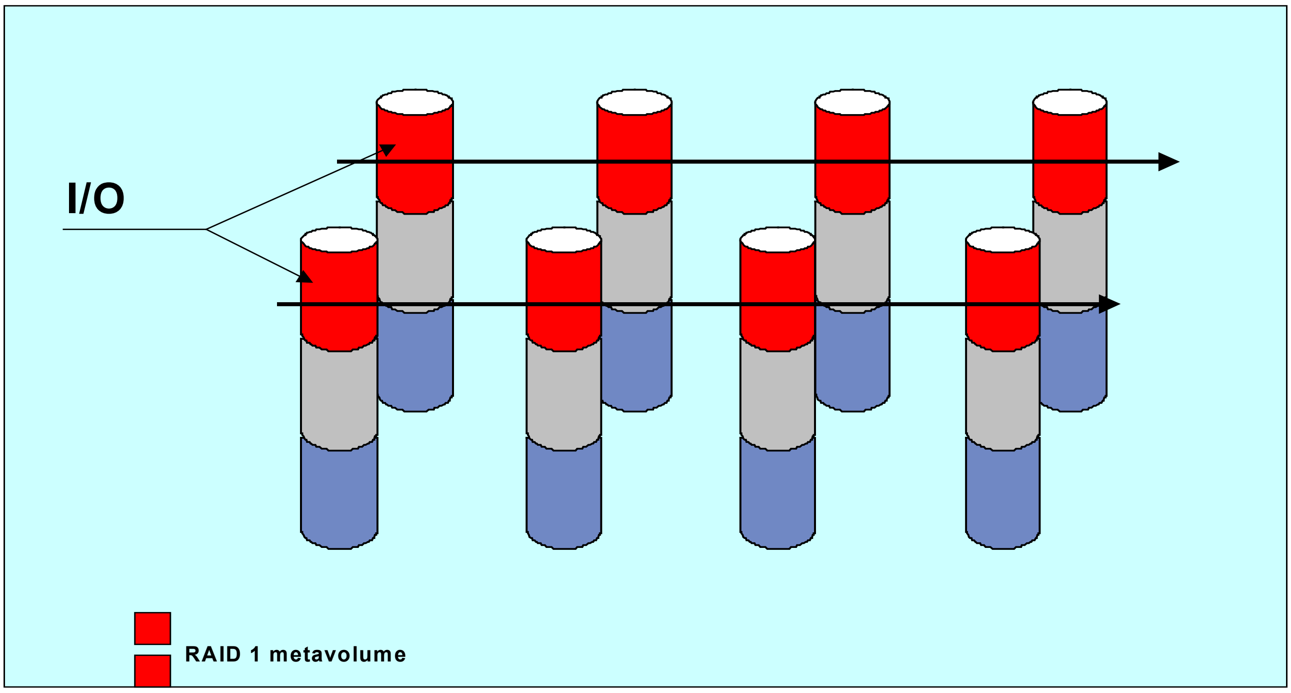
Before we can answer that question we have to understand what is happening in cache and how it integrates with the workload. A clear understanding of both will enable you to diagnose cache performance and recommend a solution.
Real workloads are composed of many different types of I/O activity. They can be read or write requests, they have different data block sizes, they can be skewed (some disks or host channels do more work than others), they can be highly random, sequential or mixed and they are often 'bursty' (peak reads or writes can come at unexpected times). The workload used for lab measurements are normally static, simple and designed to always yield certain levels of hit ratio (access of r/w data directly out of cache), regardless of the cache size and algorithms. In real life, the actual application behaviour is greatly influenced by performance optimization algorithms of the SDA.

Intelligent cache management algorithms can perform pre-fetch, LRU and FastWrite activities in order to make a read I/O a hit in cache or optimise the cache residency time. In EMC Symmetrix in addition the user has been provided the capability (Quality of Service) to define segments of cache that can be managed by a separate LRU list associated with one or more designated logical volumes. This way a user can guarantee that an important application has a minimum number of cache slots, regardless of the activity levels of the other application in the system.
Another important point is the cache parallel access. With larger and larger storage capacity, SDA has improved cache size (up to 64 GB), therefore parallel access is very important to that resource . Parallel access can reduce cache contention. The EMC Symmetrix cache is partitioned up to 16 separately addressable regions, so the contention probability that a second process is attempting to access a cache region is 6 % (1/16) or in other words it is possible to satisfy I/O requests without incurring a queue for resources, 94% of the time.
So a robust cache architecture can achieve performance improvements over other solutions and can make the difference. for a given workload in order to get higher hit ratios.

Disk configuration is another key element. Even when there are very high hit ratios (more than 90%), data must be written to disks. For example if you analyse a workload of 10.000 4K I/Os, with a 3:1 R/W ratio: 7500 will be reads and 2.500 will be writes. If we can get a 90% hit in cache, it means that 10% (1.000) of I/Os (reads) are made from disks. We have also to destage the 2.500 writes from cache to disks. Each write involves additional backend I/Os depending on RAID level protection.(we'll examine this aspect in the next paragraph and the write workload can be double or more). Now it's clear that even with a 90% of cache hits the SDA has to manage at least 1.000 + (2.500*n) I/Os where n depends on the RAID level protection. In any case the value of backend I/Os is >35% of the total (in our case > 3.500). It's very important to distribute, or 'stripe' those I/Os to different disks in order to have concurrent parallel processes. These

concepts are behind the creation inside the SDA, of volumes which are physically striped over different disks, but are presented to the operating system as they were a single physical disk (an hdisk for AIX or disk F for Windows). The mapping of these disks which reside on multiple physical disks are done by the microcode inside the SDA and are called 'metavolumes' . Each physical disk is recognized as a member in the metavolume.
The OS is not aware of this internal configuration and it only sees the 'metavolume' as a unique entity. I/O workload to that hdisk is then striped over different physical disks inside the SDA in order to improve throughput with no overhead at the CPU level. Usually 4 to 8 members are used to satisfy most application requirements especially based on Oracle or SAP.
Logical Volume Manager treats these metavolumes as all the other disks in the system therefore it's still possible to partition the metavolume into different logical volumes.
Striping could be done at the host level, like Veritas Volume Manager, but there are some disadvantages as high CPU consumptions cycles and more complex management.

The following figure depicts the Metavolume and striping architecture.



The metavolume is made by the four different coloured pieces residing on different disks. If those disks are also mirrored protected you can have an additional 4 drives to handle the back end workload for a total of 8 drives.

Different RAID level can also impact performance and data integrity protection. RAID first known as Redundant Array of Inexpensive Disks, changed its acronym to Redundant Array of Independent disks. Even if some open systems can manage software RAID it's recommended that the disk array provides this functions in order to generate less overhead at CPU and channel overhead.
Following are the description of the main RAID level implemented.

RAID 0 - it's erroneously defined raid since there is no data protection. It allows data to be written on consecutive physical drives, with a fixed number of 512-byte blocks per write.. That is, data integrity is entirely dependent on the frequency and validity of backups. This is usually known as striping and it's analogous to the Logical Volume Striping. It can be combined with some of the following RAID levels.

RAID1 - it's the simplest solution to protect data, since it consists of duplicating data across 2 or more disks (if combined with striping). Even if half of the drives are used to mirror the other drives, costs are no more an issue since larger capacity disks has become available to market. **Every**

write I/O operation from the host will exploit into two write I/O operations toward the disk backend. Microcode improvements can reduce this activity by grouping write I/Os belonging to the same track/cylinder. RAID 1 resolves all of the data integrity and availability concerns and has no performance impact in case of failure of one of the 2 disks. **In fact, in case of failure, data can be retrieved, simply reading the surving disk with no additional overhead**. There are some implemented improve techniques, like EMC mirroring, which can reduce latency time allowing read I/Os from both disks (the one better positioned to the requested block of data).

- This level of function could be analogous to the logical volume mirroring function of the logical volume manager, but do not impact on CPU activity and channel utilization (each write I/O should be duplicated across channels) and could not benefit of technique like 'mirroring'.

RAID3 - Data is striped on a byte-by-byte basis across a set of data drives, while a separate parity drive contains a parity byte for each corresponding byte position on the data drives. If any single drive fails, its contents can be inferred from the parity byte and the surviving data bytes. Usually this implementation is suited for sequential large block sized I/O; PAM (Parallel Access Method) is usually used. As it's possible to do a single I/O per time, there is no contention to the parity disk.

RAID5 - Data is striped block by (512-byte) block, but portions of several (not necessarily all) of the drives are set aside to hold parity information. This spreads the load of writing parity information more evenly, **but parity calculation adds overhead to the entire system wherever it's done. In fact every write operation from the host requires two read operations from the back end (old data and old parity) and create two additional writes (new data and new parity).** This behaviour introduces such an overhead to the backend that RAID-5 is not suggested for high performance environments like SAP and Oracle are. In addition in case of a disk failure, performance can be impacted due to data rebuild activity which involves all the disks belonging to the same RAID 5 group; in this case the more disks are configured in a RAID-5 group the more will be the I/O backend overhead in case of data reconstruction.

RAID devices should be considered primarily a data-integrity and data-availability solution, rather than a performance solution. You should also consider that higher level of RAID implementation usually involves more complex microcode management and more resources are needed.
One last item is that if performance is a concern, you should configure your raid group, whatever it is, with disks belonging to multiple SCSI adapters, rather than a single one, in order to achieve the maximum I/O back end parallelism. This requires, once again, a good knowledge of the SDA architecture.

5 Native UNIX Utilities to Analyse the I/O System

I/O performance analysis is possible by using native UNIX utilities as sar or iostat in order to collect logical disk statistics as disk busy , transfers per second and kbyte throughput per second. Other utilities such as vmstat can provide the total number of I/O's done each interval. The results of these utilities can indicate the amount of I/O throughput performed by each system. These kind of statistics need to be collected for all systems that are connected to the storage controllers. Two principle indicators of I/O activity are the amount of transfers per second performed by the system and the I/O wait percentage which indicates the percentage of time the system must wait for the processes to complete I/O activity.
One way to measure the I/O's response time can be performed by using the trace utility offered by AIX which is not discussed in this document. This writes a lot of records. Here it's important to capture the I/O functions of the program.

The AIX systems provides very few indicators that measure the I/O subsystem. In this chapter we will explain the most important measurements that can be obtained by the native utilities.
The main utilities we'll talk about are vmstat , iostat, ps and filemon. The native utility sar will not be covered because it provides the same statistics as iostat ..

The vmstat utility provides the overall system I/O for the system. The values are accumulated for each measurement. By calculating the difference between the sum of (page ins+page outs) for two consecutive intervals you get the amount of I/O which occurred in that interval. As seen in the following example the output also distinguishes between the pages paged to the file systems (page ins / outs) and the pages paged to the paging dataset (paging space page ins / outs) . The overall system I/O is calculated by means of the page ins and page outs.

An example of the vmstat –s utility.

```
 1917453168 total address trans. faults
  195652782 page ins
   35188650 page outs
        215 paging space page ins
        556 paging space page outs
          0 total reclaims
   74931412 zero filled pages faults
     323046 executable filled pages faults
  503394332 pages examined by clock
        338 revolutions of the clock hand
  201588144 pages freed by the clock
     256442 backtracks
          0 lock misses
     372144 free frame waits
          0 extend XPT waits
   32246026 pending I/O waits
   56706175 start I/Os
   56707861 iodones
  784710926 cpu context switches
  345259597 device interrupts
          0 software interrupts
          0 traps
  265340360 syscalls
```

The iostat provides statistics for each disk available on the system . Measurement statistics as kbytes transfer per second (kbps) , disk busy (% _tm_act) , transactions per second (tps) , kbytes read (kb_read) , kbytes written l (kb_wrtn) , percent of time the system had to wait to perform I/O (% iowait).
The interval length and frequency are parameters provided to the utility. The utility does not provide a timestamp for the interval written. This must be done by developing a homemade script.

An example of the output of iostat :

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.2         25.8              18.9      8.5       66.0       6.6

Disks:         % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0            3.1      26.1      4.6    2542641   7014128
hdisk1            2.9      20.1      4.2     342724   7015280
hdisk137          0.0       0.0      0.0          0         0
hdisk138          0.0       0.0      0.0          0         0
hdisk139          0.0       0.0      0.0          0         0
hdisk140          0.0       0.0      0.0          0         0
hdisk141          0.0       0.0      0.0          0         0
hdisk142          0.0       0.0      0.0          0         0
hdisk143          0.0       0.0      0.0          0         0
hdisk144          0.0       0.0      0.0          0         0
hdisk145          0.0       0.0      0.0          0         0
hdisk146          0.0       0.0      0.0          0         0
cd0               0.0       0.0      0.0          0         0
```

All the disks seen by iostat are defined as available to the operating system. Some disks are attached internally to the server while others are attached externally. The I/O configuration data can be displayed in the AIX environment by the lscfg utility. The output of this utility displays configuration, diagnostic, and vital product data information about the system. It provides then name , locations and descriptions of each device configured. In the following example I extracted some disks and adapters.

The lscfg utility just provides the devices defined to the system.

Partial output of the **lscfg** utilty

```
Name                location            description
+ scsi0             10-60               Wide/Fast-20 SCSI I/O Controller
+ ses0              10-60-00-15,0       SCSI Enclosure Services Device
+ scsi1             10-68               Wide/Fast-20 SCSI I/O Controller
+ cd0               10-68-00-3,0        SCSI Multimedia CD-ROM Drive (650
                                        MB)
* pci1              00-f8500000         PCI Bus
+ sa2               20-58               IBM 8-Port EIA-232/RS-422A (PCI)
                                        Adapter
+ ssa0              20-68               IBM SSA 160 SerialRAID Adapter
                                        (14109100)
+ ssa1              30-58               IBM SSA 160 SerialRAID Adapter
                                        (14109100)
+ scsi2             30-60               Wide/Fast-20 SCSI I/O Controller
+ hdisk137          30-60-00-0,0        EMC Symmetrix SCSI RDF1 RaidS
+ hdisk138          30-60-00-0,1        EMC Symmetrix SCSI RDF1 RaidS
+ hdisk184          50-68-00-15,0       EMC Symmetrix SCSI RaidS
+ hdisk185          50-68-00-15,1       EMC Symmetrix SCSI RaidS
+ scsi5             50-70               Wide/Fast-20 SCSI I/O Controller
+ fcs1              60-58               FC Adapter
* fscsi1            60-58-01            FC SCSI I/O Controller Protocol
                                        Device
+ scsi6             60-68               Wide/Fast-20 SCSI I/O Controller
+ ent4              60-70               IBM 10/100 Mbps Ethernet PCI Adapter
                                        (23100020)
* pci6              00-f8e00000         PCI Bus
```

```
+ scsi7             70-58             Wide/Fast-20 SCSI I/O Controller
+ scsi8             70-60             Wide/Fast-20 SCSI I/O Controller
+ scsi9             70-68             Wide/Fast-20 SCSI I/O Controller
+ scsi10            70-70             Wide SCSI I/O Controller
* pci7              00-f8f00000       PCI Bus
* hdisk2            20-68-L           SSA Logical Disk Drive
* hdisk3            20-68-L           SSA Logical Disk Drive
* hdisk234          20-68-L           SSA Logical Disk Drive
* hdisk214          20-68-L           SSA Logical Disk Drive
+ pdisk9            20-68-D40C-01-P   SSA160 Physical Disk Drive (18200
                                      MB)
+ pdisk4            20-68-D40C-02-P   SSA160 Physical Disk Drive (18200
                                      MB)
+ pdisk1            20-68-D40C-03-P   SSA160 Physical Disk Drive (18200
                                      MB)
```

The lsdev utility shows the devices that are available to the system.
You can notice the available and defined devices in the following example.

A partial output of the lsdev utility

```
scsi0           Available 10-60           Wide/Fast-20 SCSI I/O Controller
scsi1           Available 10-68           Wide/Fast-20 SCSI I/O Controller
isa0            Available 10-78           ISA Bus
ssa0            Available 20-68           IBM SSA 160 SerialRAID Adapter
ssa1            Available 30-58           IBM SSA 160 SerialRAID Adapter
scsi2           Available 30-60           Wide/Fast-20 SCSI I/O Controller
ssa2            Available 30-70           IBM SSA 160 SerialRAID Adapter
fcs0            Available 50-58           FC Adapter
scsi4           Available 50-68           Wide/Fast-20 SCSI I/O Controller
scsi5           Available 50-70           Wide/Fast-20 SCSI I/O Controller
fcs1            Available 60-58           FC Adapter
ent3            Available 60-60           IBM 10/100 Mbps Ethernet PCI Adapter
scsi6           Available 60-68           Wide/Fast-20 SCSI I/O Controller
ent4            Available 60-70           IBM 10/100 Mbps Ethernet PCI Adapter
scsi7           Available 70-58           Wide/Fast-20 SCSI I/O Controller
ses0            Available 10-60-00-15,0   SCSI Enclosure Services Device
cd0             Available 10-68-00-3,0    SCSI Multimedia CD-ROM Drive
hdisk0          Available 30-68-00-8,0    16 Bit LVD SCSI Disk Drive
hdisk1          Available 30-68-00-9,0    16 Bit LVD SCSI Disk Drive
ses1            Available 30-68-00-15,0   SCSI Enclosure Services Device
fscsi0          Available 50-58-01        FC SCSI I/O Controller Protocol Device
fscsi1          Available 60-58-01        FC SCSI I/O Controller Protocol Device
lvdd            Available                 LVM Device Driver
ssar            Defined                   SSA Adapter Router
pdisk0          Available 20-68-D40C-07-P SSA160 Physical Disk Drive
pdisk1          Available 20-68-D40C-03-P SSA160 Physical Disk Drive
hdisk106        Available 70-58-00-2,3    EMC Symmetrix SCSI RDF1 RaidS
hdisk107        Available 70-58-00-2,4    EMC Symmetrix SCSI RDF1 RaidS
hdisk223        Defined   30-60-00-2,4    EMC Symmetrix SCSI RDF1 RaidS
hdisk224        Defined   30-60-00-2,5    EMC Symmetrix SCSI RDF1 RaidS
```

The ps utility provides an options to analyse the amount of I/O done on behalf of the processes :

PGIN - (**v** flag option) The number of disk I/Os resulting from references by the process to pages not loaded in core.

In the following you'll find an example of ps command output:

```
ps vgcw

       PID              PGIN         %MEM          COMMAND
         0                 7          1.0          swapper
         1               130          0.0             init
      3748              1816          0.0            syncd
      5722                 1          0.0        ssa_daemo
      7282               147          0.0         errdemon
     16792                 2          0.0       rpc.lockd
     23704                14          0.0              ksh
     26322                26          0.0           vmstat
     26634                 9          0.0            eqqtw
```

5.1 FILEMON STATISTICS

Monitors the performance of the file system, and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes.

Logical file system
The **filemon** command monitors logical I/O operations on logical files. The monitored operations include all **read**, **write**, **open**, and **lseek** system calls, which may or may not result in actual physical I/O, depending on whether or not the files are already buffered in memory. I/O statistics are kept on a per-file basis. Virtual memory system.

The **filemon** command monitors physical I/O operations (that is, paging) between segments and their images on disk. I/O statistics are kept on a per-segment basis. Logical volumes

The **filemon** command monitors I/O operations on logical volumes. I/O statistics are kept on a per-logical-volume basis. Physical volumes

The **filemon** command monitors I/O operations on physical volumes. At this level, physical resource utilizations are obtained. I/O statistics are kept on a per-physical-volume basis.

Most Active Logical Volumes Report

Column	Description
`util`	Utilization of the volume (fraction of time busy). The rows are sorted by this field, in decreasing order.
`#rblk`	Number of 512-byte blocks read from the volume.
`#wblk`	Number of 512-byte blocks written to the volume.
`KB/sec`	Total transfer throughput, in Kilobytes per second.
`volume`	Name of volume.
`description`	Contents of volume: either a file system name, or logical volume type (paging, jfslog, boot, or sysdump). Also, indicates if the file system is fragmented or compressed.

Most Active Physical Volumes Report

Column	Description
`util`	Utilization of the volume (fraction of time busy). The rows are sorted by this field, in decreasing order.
`#rblk`	Number of 512-byte blocks read from the volume.
`#wblk`	Number of 512-byte blocks written to the volume.
`KB/sec`	Total volume throughput, in Kilobytes per second.
`volume`	Name of volume.
`description`	Type of volume, for example, `120MB disk`, `355MB SCSI`, or `CDROM SCSI`.

Note: Logical volume I/O requests start before, and end after, physical volume I/O requests. For that reason, total logical volume utilization will appear to be higher than total physical volume utilization.

By looking at the location it's possible to see what scsi locations the disk is **`Using.`**

```
Scsi10           Available 30-60          Wide/Fast-20 SCSI I/O Controller
Hdisk900         Available 30-60-00-2,4   EMC Symmetrix SCSI RDF1 RaidS
```

The disk hdisk900 is located on scsi adapter 30-60.

The command **lsdev -Cs scsi** reports on the current address assignments on each SCSI bus. For the original SCSI adapter, the SCSI address is the first number in the fourth pair of numbers in the output. In the following output example, the 400MB disk is at SCSI address 0, the 320MB disk at address 1, and the 8mm tape drive at address 5.

```
hdisk0   Available 00-01-00-00 400 MB SCSI Disk Drive
hdisk1   Available 00-01-00-10 320 MB SCSI Disk Drive
rmt0     Defined   00-01-00-50 2.3 GB 8mm Tape Drive
```

6 ROTS - rules of thumb

Some points to acknowledge before considering ROTS.

1) The higher the blocksize the higher the bus utilization with less protocol overhead. The protocol traffic used is about 50% of the bandwidth of the bus for small blocksize I/O's.
2) the average I/O size performed by the application, how many applications will run concurrently
3) the kind of channel available ? Fast Wide SCSI, Ultra-SCSI, FC
4) kind of raid configuration do I need? Raid 10 (mirroring plus striping) is better than Raid 5?
5) striping or not striping?
6) the average utilization of the disks (the maximum I/O that can be performed by a disk is about 100 I/O's second) if the average utilization is 20% then the average I/O's would be also 20 I/O's second. Based on the bus's throughput capacity and number of I/O's it's possible to calculate the number of disks you need to held your workload.
7) Always look for the slowest component in the I/O physical path.
8) Simulation tools from Disk Array vendors and cache hit analysis can help you in configuring your disk subsystem.

6.1 Channel bandwith and blocksize

Studies of both open systems and MVS systems show that average utilization for medium- to large-scale disk farms is 20% or less, with very large systems averaging only 8%. Assuming 20% average utilization (and thus about 20 I/Os per second per disk drive), the maximum number of drives per bus looks like this:

	I/O Size			
Bus speed	2K	4K	8K	16K
fast	126	82	48	26
fast/wide	173	126	82	48
fast-20/wide	212	173	126	82
fast-40/wide	240	212	173	126

5000/2= 2500 I/O of 2K

Bus capacity/ throughtput_sec 5000k/2k =2500 I/O seconds

5000/2= 2500 I/O 2500/(100*.20)= 125 disks (20% = 20 tps)

5000/2= 2500 I/O 2500/100= 25 disks (100% = 100 tps)

6.2 Access Density

Access density is defined as the total throughput divided by the total space. Based on many experiences done at different sites the value of 0.2 seems to provide the correct relationship. Based on this knowledge it could be possible for us to calculate the amount of I/O the system will perform or how much space we'll need.

7 Storage Vendor Utilities

First signs of high disk busy, high queuing (not all systems provide queuing) could point out that disks are suffering from I/O problems. At this stage we would have to locate the logical disks physical positions inside the storage controller and check if other logical disks from the same or other systems are located on these same physical disks. The matching of the physical locations and further performance statistics at the physical and logical level (%write, %read, I/O sec, MB/s) or for the entire storage controller (channel and cache utilization, channels, ports) can be done by storage controller manufacturer vendors

By using the an EMC utility called syminq it's possible to get the list of the defined disks on the system. This utility performs an inquiry on all the scsi disks in the system and shows their characteristics.

The column description are as follows :

Name	- name of device
Type	- configurations type (EMC disks provide additional info as SRDF, gatekeeper,etc)
Vendor	- (EMC , IBM , etc)
ID	- device identification
REV	- revision (microcode version)
Ser Num	-the order from right to left last two bytes - symmmetrix storage disk array. three middle bytes - the logical volume (sym device) first two bytes- the port number (port A=0-16 , port B=17-29)
Cap	- capacity

```
     Device                        Product              Device
------------------- --------- --------------------- ------------------
     Name      Type  Vendor    ID                Rev  Ser Num    Cap (KB)
------------------- --------- --------------------- ------------------

/dev/rhdisk0        IBM       DMVS09V           3031 F80383F2       N/A
/dev/rhdisk1        IBM       DMVS09V           3031 F803CBDC       N/A
/dev/rhdisk2        IBM       SSA DEVICE                            N/A
/dev/rhdisk3        IBM       SSA DEVICE                            N/A
/dev/rhdisk17  R1   EMC       SYMMETRIX         5265 6300D291       N/A

EXAMPLE:
63  - SYMMETRIX
00D - (LOGICAL VOLUME) SYM DEVICE
29 - Processor B
1 Port B

/dev/rhdisk18  R1   EMC       SYMMETRIX         5265 6300E291       N/A
/dev/rhdisk19  R1   EMC       SYMMETRIX         5265 6300F291       N/A
/dev/rhdisk21  R1   EMC       SYMMETRIX         5265 63011291       N/A
/dev/rhdisk118 R1   EMC       SYMMETRIX         5265 63072271       N/A
/dev/rhdisk119 R1   EMC       SYMMETRIX         5265 63073271       N/A
/dev/rhdisk184 GK   EMC       SYMMETRIX         5265 63090291       N/A
/dev/rhdisk185 GK   EMC       SYMMETRIX         5265 63091291       N/A
/dev/rhdisk186 GK   EMC       SYMMETRIX         5265 63092291       N/A
/dev/rhdisk187 R1   EMC       SYMMETRIX         5265 63001291       N/A
/dev/rhdisk189 R1   EMC       SYMMETRIX         5265 63003291       N/A
/dev/rhdisk232 R1   EMC       SYMMETRIX         5265 6301A271       N/A
/dev/rhdisk233 R1   EMC       SYMMETRIX         5265 6301B271       N/A
/dev/rhdisk234      IBM       SSA DEVICE                            N/A
/dev/rhdisk295 GK   EMC       SYMMETRIX         5265 63090201       N/A
/dev/rhdisk296 GK   EMC       SYMMETRIX         5265 63091201       N/A
```

7.1 Control Center Open Edition (CCOE)

Control Center Open Edition is the base software that allows you to interact with the SDA and discover the mapping of logical volumes to physical volumes. It has a multi-tier architecture with host(s) at each tier running different CC components that perform specific roles. All tiers can co-exist on a single host (single host infrastructure) or they can be distributed to several hosts (distributed infrastructure).
The architecture of the CCOE component can be found in appendix A.
We will examine some of the CC components which are best related to our study, like Symmetrix Manager, Workload Analyzer and Optimizer.

7.2 CC Symmetrix Manager

This component of CCOE allows realtime monitoring of SDA status and performance. It makes it possible to understand if the I/O workload on that disk exceeds the number of I/O's the disk can handle, if there are resource contentions inside the SDA (for example at the director level). When a performance problem arises, the administrator can drill down and find out which is the physical disk involved, how many logical volumes are mapped to that disk, which filesystem is allocated on a specific logical volume. Through relationship tables it's possible to relate the database, tablespace, DB instance, DB file, Filesystem, Volume Group, Logical Volume, Host Device, Host to Symmetrix front-end director and port, Symmetrix device, Symmetrix disk.
In addition Symmetrix Manager can create new logical devices, change logical device types, create or modify meta devices, set host director port attributes.
It is also possible to set thresholds and alerts and integrate the components into an existent framework like Tivoli, Patrol, Unicenter.

7.3 CC Workload Analyzer

This component of CCOE tracks and keeps historical info about performance behaviour of each single component inside the SDA. Charts are available for the following components:
Channel interface (MB/s); cache hits for all I/O types; read/write ratio; logical volumes I/O workload, etc…
Once you know the relationship between the physical and logical volumes, you can understand if there are overloaded disks when compared one to another. In addition it's possible to create capacity plan charts in order to provide the correct dimension for the future architecture.
A sample of total throughput from a specific server is shown. It is now easy to understand if channel utilization is a potential bottleneck and if we need additional channels from the server to access the SDA.

With Workload Analyzer it is therefore possible to monitor and discover all potential bottlenecks inside the I/O chain from the channels up to the physical disks. It's also possible to monitor if I/O striping is well balanced across all the physical volumes belonging to the logical striped volume.

7.4 CC Optimizer

The CCOE component allows you to automatically balance the I/O workload across the disks inside the SDA. Optimizer gathers all statistics (number of I/Os, MB/s transferred, etc…) for a specified group of volumes in a specified interval. It then creates a list of the highest disks accessed in order to swap the physical disks data with each other in order to balance the I/O workload. There should be relief from the day by day tuning to find out performance bottlenecks inside the SDA.

8 Conclusions

Native UNIX tools provide a limited view of the storage disk arrays. The results of these utilities can be used as an initial step when performing storage array analysis but they are not sufficient.
There is a need to understand the internal storage architecture (disks, cache, stripping schemes), attached components (adapters, buses , channels, remote links) and software components of the Storage Disk Array which will permit us to gather SDA data and go in deeper detail.

We hope this document will help you when performing disk storage performance analysis in the Open environment .

9 APPENDIX A – EMC utilities' architecture and graphs

9.1 Control Center Open Edition architecture

The first tier of CC contains the **Console(s)** and other CC applications; the Console is the user interface to the CC through where the user can perform all tasks.

The second tier of ECC contains the infrastructure components: the **ECC Server** (for common services, like Console data requests, user control access to specific CC objects, tracking of all managed objects and their communications); **the Repository** (contains both historical and current information on configuration, status, health data; it is the core intelligence of the system); **Performance Archive(s)** (historical Symmetrix and Host performance data); the **Store(s)** (which is responsible for converting data coming from the Agents into the relational tables of the Repository).

The third tier is the **Agents** tier. It consists of two different kinds of Agents: a **Master Agent** which runs on each Host that manages the **Individual Agents** which manage or monitor specific object domains, such as Storage Agents, Host Agents, Database Agents, Connectivity Agents. Agents pass the data they collect to the Store, which writes it to the Repository or can collect transient data, such as alerts and real time performance data, and pass this directly to the CC Server.

We will examine some of the CC components which are best related to our study, like Symmetrix Manager, Workload Analyzer and Optimizer.

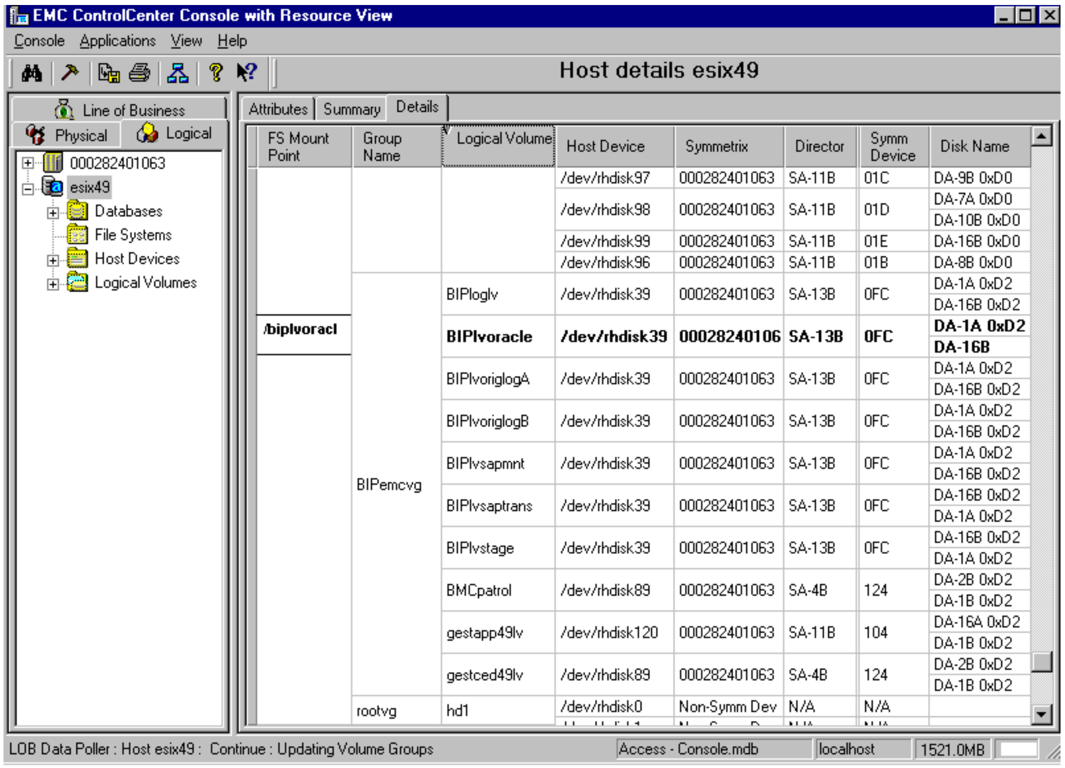
9.2 Symmetrix Manager Utilization example

The following charts displays an example of how to use this tool.

If there is a performance problem on a filesystem or if I want to investigate a specific logical volume, using Symmetrix Manager I can drill down to discover, for example, on which hdisk the filesystem is allocated (1-2).
Once we have found the hdisk, I can map it to the physical disks it resides on (2). I gather info on target and LUN ID, the SDA unit serial number, its channel adapter to which the server is connected to, etc…

At this point it's clear that I can have all the unique information regarding the internal and external physical configuration.

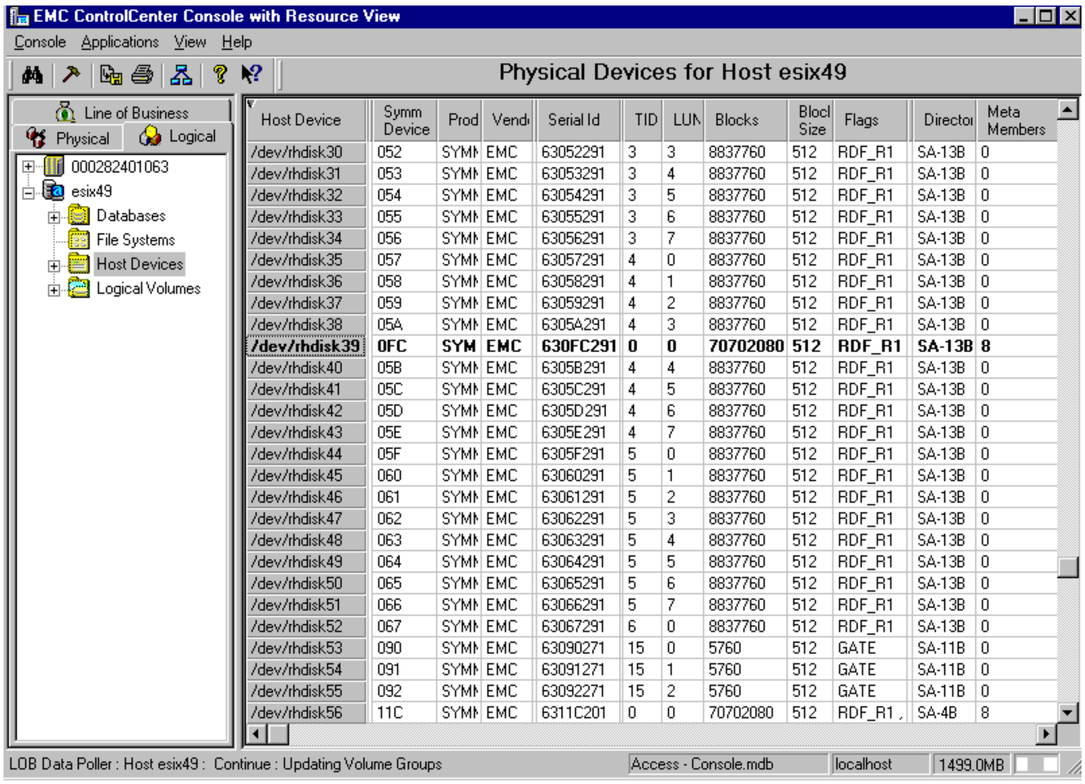
The following charts show the exact sequence I get when operating with Symmetrix Manager.



FS Mount Point	Group Name	Logical Volume	Host Device	Symmetrix	Director	Symm Device	Disk Name
			/dev/rhdisk97	000282401063	SA-11B	01C	DA-9B 0xD0
			/dev/rhdisk98	000282401063	SA-11B	01D	DA-7A 0xD0 DA-10B 0xD0
			/dev/rhdisk99	000282401063	SA-11B	01E	DA-16B 0xD0
			/dev/rhdisk96	000282401063	SA-11B	01B	DA-8B 0xD0
		BIPloglv	/dev/rhdisk39	000282401063	SA-13B	0FC	DA-1A 0xD2 DA-16B 0xD2
/biplvoracl		**BIPlvoracle**	**/dev/rhdisk39**	**00028240106**	**SA-13B**	**0FC**	**DA-1A 0xD2 DA-16B**
		BIPlvoriglogA	/dev/rhdisk39	000282401063	SA-13B	0FC	DA-1A 0xD2 DA-16B 0xD2
		BIPlvoriglogB	/dev/rhdisk39	000282401063	SA-13B	0FC	DA-1A 0xD2 DA-16B 0xD2
	BIPemcvg	BIPlvsapmnt	/dev/rhdisk39	000282401063	SA-13B	0FC	DA-1A 0xD2 DA-16B 0xD2
		BIPlvsaptrans	/dev/rhdisk39	000282401063	SA-13B	0FC	DA-16B 0xD2 DA-1A 0xD2
		BIPlvstage	/dev/rhdisk39	000282401063	SA-13B	0FC	DA-16B 0xD2 DA-1A 0xD2
		BMCpatrol	/dev/rhdisk89	000282401063	SA-4B	124	DA-2B 0xD2 DA-1B 0xD2
		gestapp49lv	/dev/rhdisk120	000282401063	SA-11B	104	DA-16A 0xD2 DA-1B 0xD2
		gestced49lv	/dev/rhdisk89	000282401063	SA-4B	124	DA-2B 0xD2 DA-1B 0xD2
	rootvg	hd1	/dev/rhdisk0	Non-Symm Dev	N/A	N/A	

LOB Data Poller : Host esix49 : Continue : Updating Volume Groups | Access - Console.mdb | localhost | 1521.0MB

(1)


EMC ControlCenter Console with Resource View
Console Applications View Help
Physical Devices for Host esix49
Line of Business
Physical
Logical
000282401063
esix49
Databases
File Systems
Host Devices
Logical Volumes

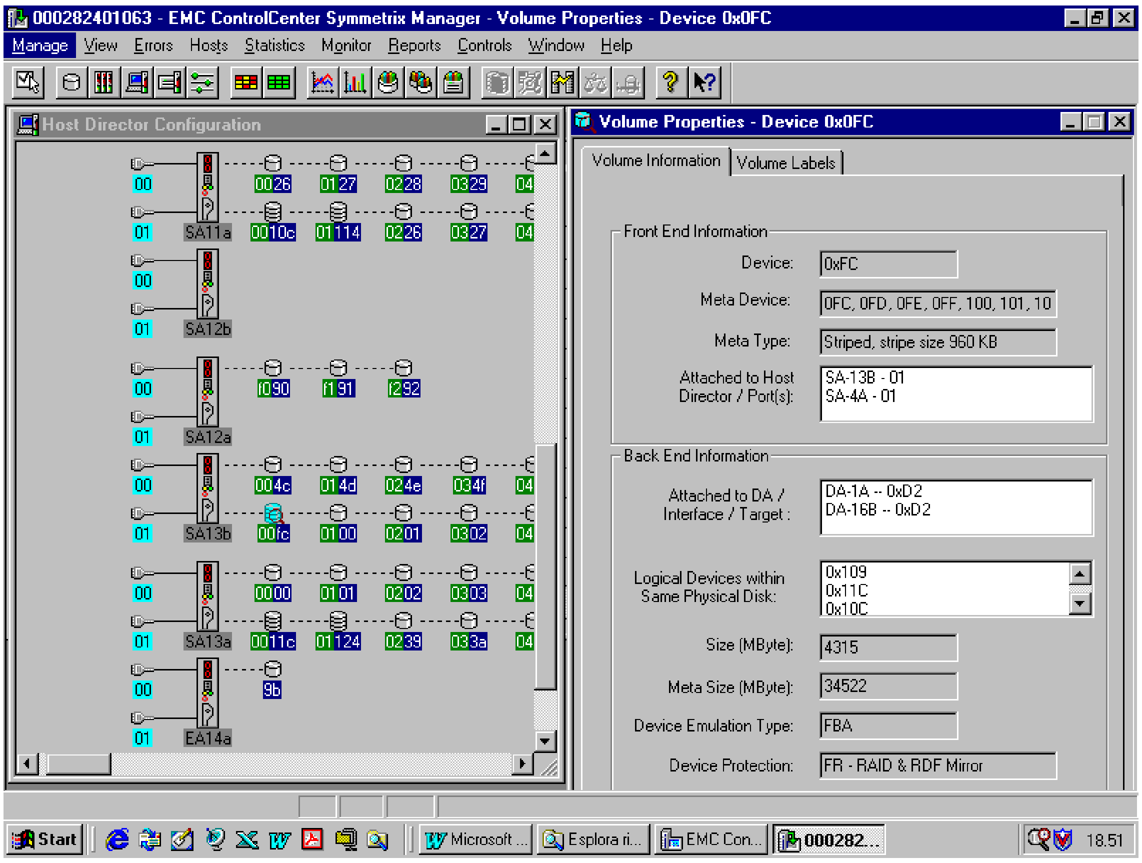
Host Device	Symm Device	Prod	Vend	Serial Id	TID	LUN	Blocks	Block Size	Flags	Director	Meta Members
/dev/rhdisk30	052	SYM	EMC	63052291	3	3	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk31	053	SYM	EMC	63053291	3	4	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk32	054	SYM	EMC	63054291	3	5	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk33	055	SYM	EMC	63055291	3	6	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk34	056	SYM	EMC	63056291	3	7	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk35	057	SYM	EMC	63057291	4	0	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk36	058	SYM	EMC	63058291	4	1	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk37	059	SYM	EMC	63059291	4	2	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk38	05A	SYM	EMC	6305A291	4	3	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk39	**0FC**	**SYM**	**EMC**	**630FC291**	**0**	**0**	**70702080**	**512**	**RDF_R1**	**SA-13B**	**8**
/dev/rhdisk40	05B	SYM	EMC	6305B291	4	4	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk41	05C	SYM	EMC	6305C291	4	5	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk42	05D	SYM	EMC	6305D291	4	6	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk43	05E	SYM	EMC	6305E291	4	7	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk44	05F	SYM	EMC	6305F291	5	0	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk45	060	SYM	EMC	63060291	5	1	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk46	061	SYM	EMC	63061291	5	2	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk47	062	SYM	EMC	63062291	5	3	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk48	063	SYM	EMC	63063291	5	4	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk49	064	SYM	EMC	63064291	5	5	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk50	065	SYM	EMC	63065291	5	6	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk51	066	SYM	EMC	63066291	5	7	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk52	067	SYM	EMC	63067291	6	0	8837760	512	RDF_R1	SA-13B	0
/dev/rhdisk53	090	SYM	EMC	63090271	15	0	5760	512	GATE	SA-11B	0
/dev/rhdisk54	091	SYM	EMC	63091271	15	1	5760	512	GATE	SA-11B	0
/dev/rhdisk55	092	SYM	EMC	63092271	15	2	5760	512	GATE	SA-11B	0
/dev/rhdisk56	11C	SYM	EMC	6311C201	0	0	70702080	512	RDF_R1 ,	SA-4B	8

LOB Data Poller : Host esix49 : Continue : Updating Volume Groups | Access - Console.mdb | localhost | 1499.0MB

(2)

We could go deeper and find out the other internal info; in this example we can see that the selected hdisk is allocated over an 8 members metavolume, the hdisk is shared between two channel interfaces (SA-13B and SA 4A). Moreover on the same physical disk there are other Symmetrix device (10C, 11C, 109, etc…)

which are allocated to other hdisks and therefore to filesystems sharing the same physical disk (3-4). In case of performance degradation it is easy to find out potential bottlenecks.


000282401063 - EMC ControlCenter Symmetrix Manager - Volume Properties - Device 0x0FC
Manage View Errors Hosts Statistics Monitor Reports Controls Window Help
Host Director Configuration
SA11a
SA12b
SA12a
SA13b
SA13a
EA14a
Volume Properties - Device 0x0FC
Volume Information
Volume Labels
Start
18.51

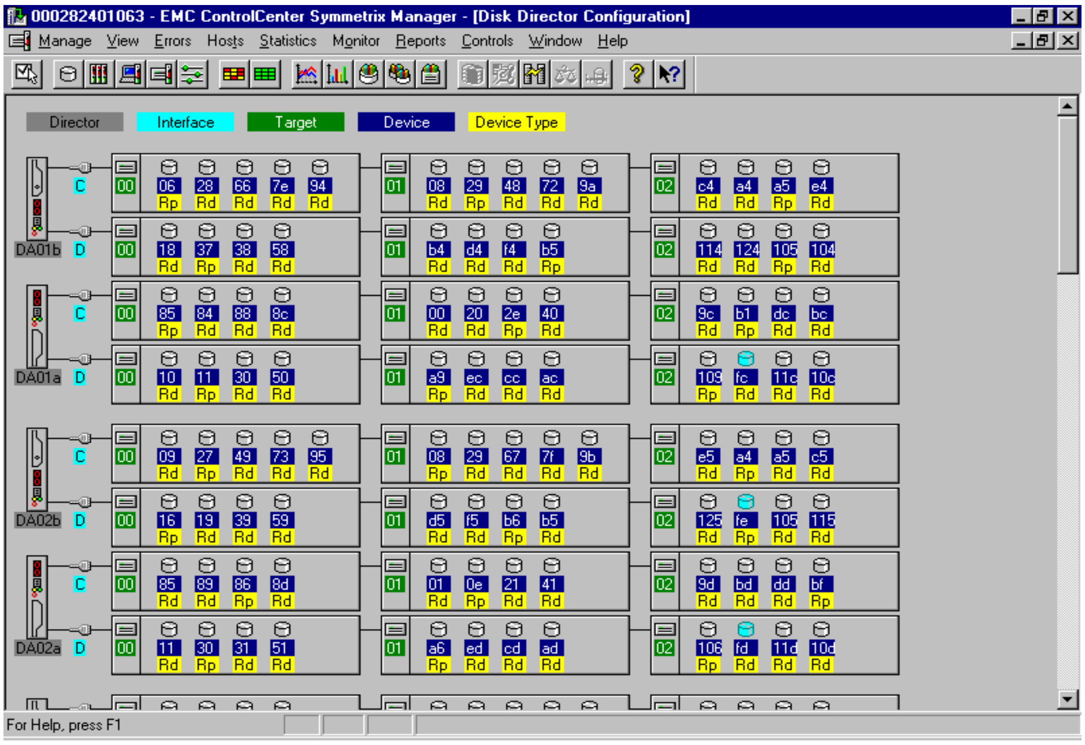
Front End Information

Device:	0xFC
Meta Device:	0FC, 0FD, 0FE, 0FF, 100, 101, 10
Meta Type:	Striped, stripe size 960 KB
Attached to Host Director / Port(s):	SA-13B - 01 SA-4A - 01

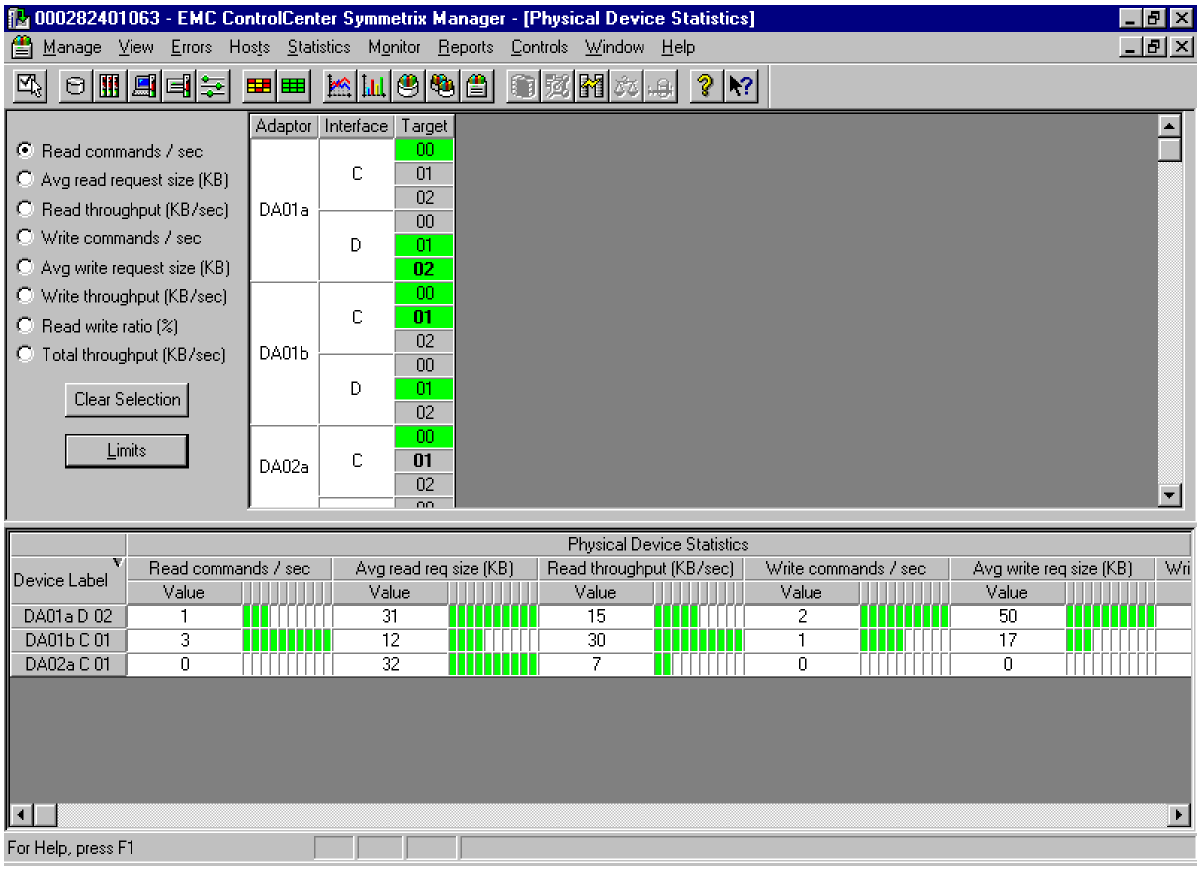
Back End Information

Attached to DA / Interface / Target :	DA-1A -- 0xD2 DA-16B -- 0xD2
Logical Devices within Same Physical Disk:	0x109 0x11C 0x10C
Size (MByte):	4315
Meta Size (MByte):	34522
Device Emulation Type:	FBA
Device Protection:	FR - RAID & RDF Mirror

(3)


000282401063 - EMC ControlCenter Symmetrix Manager - [Disk Director Configuration]
Manage View Errors Hosts Statistics Monitor Reports Controls Window Help
Director
Interface
Target
Device
Device Type
DA01b
DA01a
DA02b
DA02a
For Help, press F1

(4)


000282401063 - EMC ControlCenter Symmetrix Manager - [Physical Device Statistics]
Manage View Errors Hosts Statistics Monitor Reports Controls Window Help
For Help, press F1

[✓ Read commands / sec]
Avg read request size (KB)
Read throughput (KB/sec)
Write commands / sec
Avg write request size (KB)
Write throughput (KB/sec)
Read write ratio (%)
Total throughput (KB/sec)

Clear Selection

Limits

Adaptor	Interface	Target
DA01a	C	🟩 00
		01
		02
	D	00
		🟩 01
		🟩 **02**
DA01b	C	🟩 00
		🟩 **01**
		02
	D	00
		🟩 01
		02
DA02a	C	🟩 00
		01
		02

Physical Device Statistics

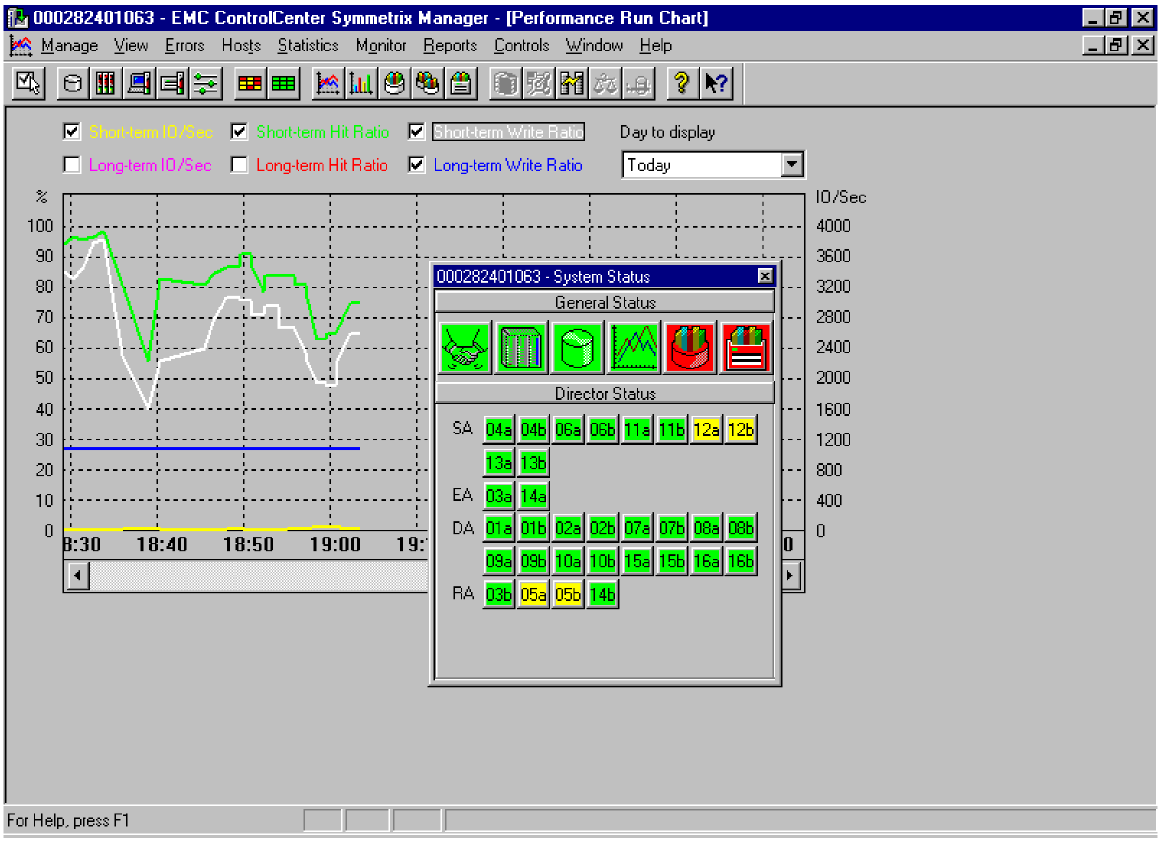
Device Label	Read commands / sec	Avg read req size (KB)	Read throughput (KB/sec)	Write commands / sec	Avg write req size (KB)	Wri
	Value	Value	Value	Value	Value	
DA01a D 02	1	31	15	2	50	
DA01b C 01	3	12	30	1	17	
DA02a C 01	0	32	7	0	0	

(5)

I can also monitor the read/write ratio, the average I/O size, the total throughput, etc… (chart 5)

If predefined alert thresholds are reached then specific actions can be taken from the optional exiting frameworks like Tivoli, Unicenter, etc....

The following chart (6) displays online monitoring of the cache hit ratios, the read/write ratios and the I/O rate.

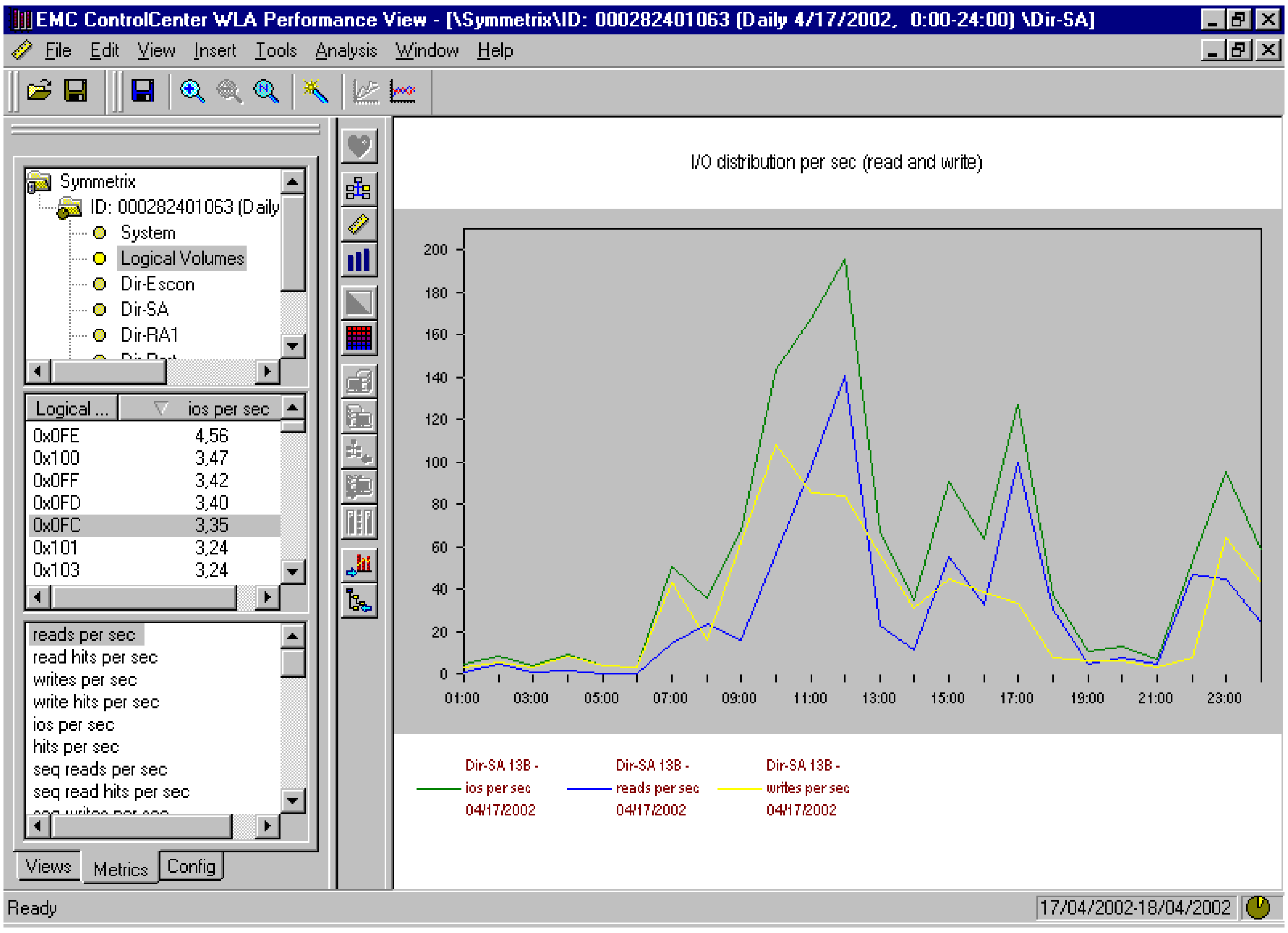

000282401063 - EMC ControlCenter Symmetrix Manager - [Performance Run Chart]
Manage View Errors Hosts Statistics Monitor Reports Controls Window Help
Short-term IO/Sec
Short-term Hit Ratio
Short-term Write Ratio
Long-term IO/Sec
Long-term Hit Ratio
Long-term Write Ratio
Day to display
Today
%
100
90
80
70
60
50
40
30
20
10
0
IO/Sec
4000
3600
3200
2800
2400
2000
1600
1200
800
400
0
8:30
18:40
18:50
19:00
19:
000282401063 - System Status
General Status
Director Status
SA 04a 04b 06a 06b 11a 11b 12a 12b 13a 13b
EA 03a 14a
DA 01a 01b 02a 02b 07a 07b 08a 08b 09a 09b 10a 10b 15a 15b 16a 16b
RA 03b 05a 05b 14b
For Help, press F1

(6)

9.3 Workload Analyzer

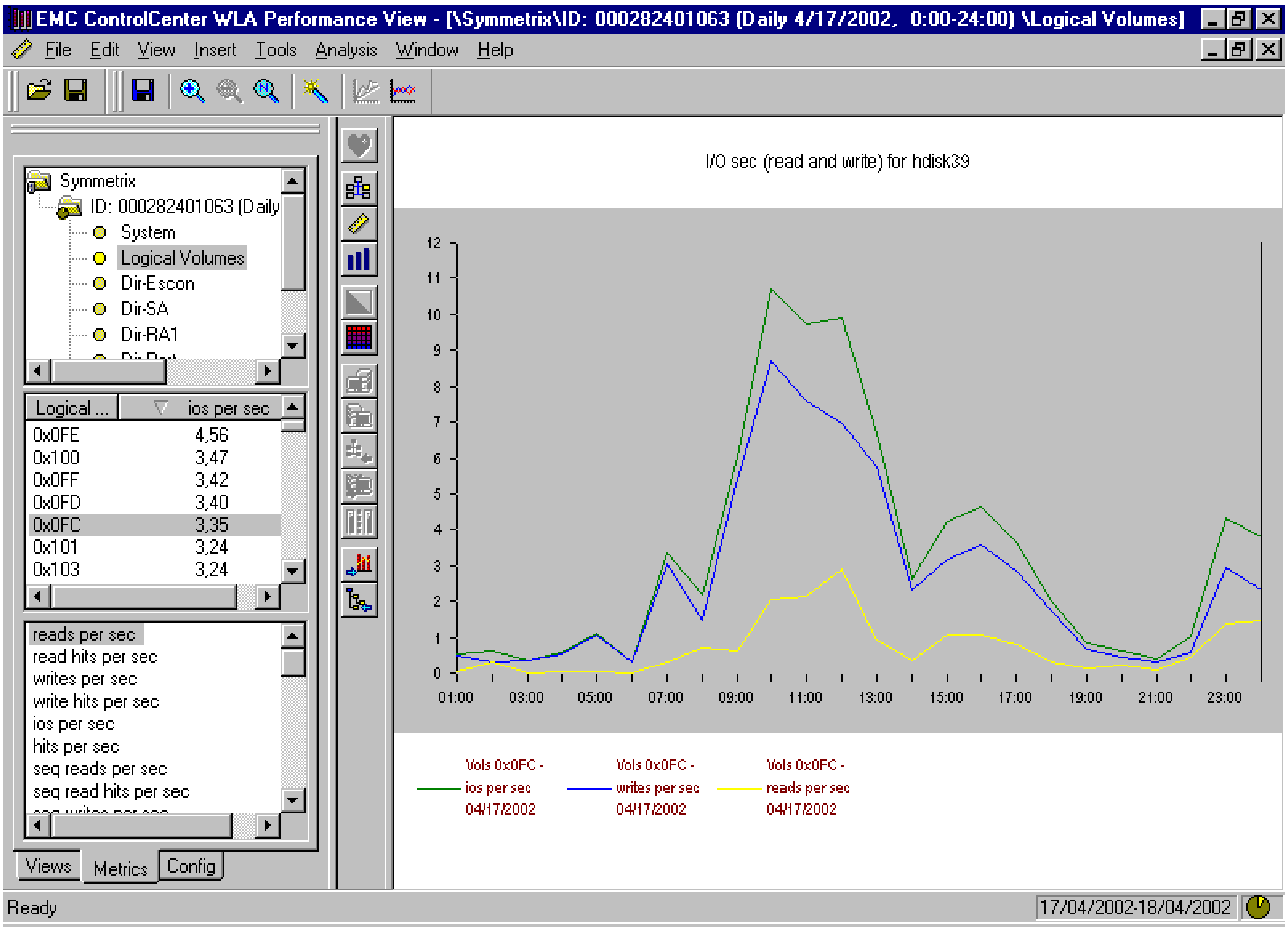
In the example below we are monitoring the % utilization of all channels from ESIX49 server to Symmetrix SDA. As you can see the utilization values are far below (5%) any alert threshold (suggested values 40-50%). This graph can be useful for capacity plan activity when dimension the number of paths a server needs to access disks.

In the example below we are monitoring the total I/O per sec and reads and writes flow through DIR 13B Symmetrix channel interface. From this graph we can understand the workload which is generated on this interface. In the next graph we can go deeper to the single logical volume.



Logical ...	ios per sec
0x0FE	4,56
0x100	3,47
0x0FF	3,42
0x0FD	3,40
0x0FC	3,35
0x101	3,24
0x103	3,24

In the example below we are monitoring the total I/O per sec and reads and writes of raw device hdisk39 configured on Symmetrix device 0FC.

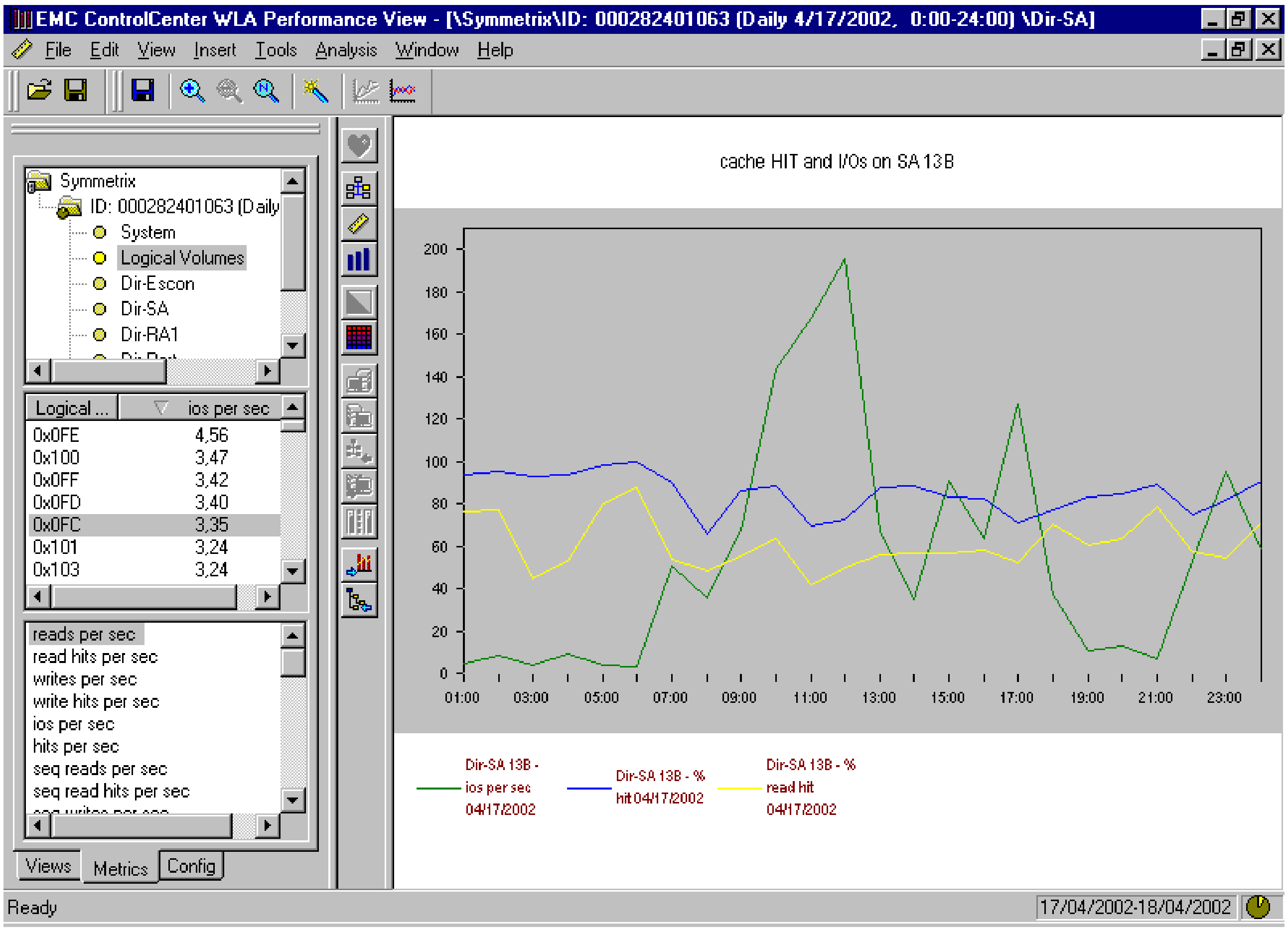


EMC ControlCenter WLA Performance View - [\Symmetrix\ID: 000282401063 [Daily 4/17/2002, 0:00-24:00] \Logical Volumes]

Logical ...	ios per sec
0x0FE	4,56
0x100	3,47
0x0FF	3,42
0x0FD	3,40
0x0FC	3,35
0x101	3,24
0x103	3,24

The specific file system is mounted on this raw device.
All those values can be useful when we need info such read/write ratios, or total I/Os over a specific hdisk (or filesystem)

This window shows the % hit in cache, the % read hits and the total I/Os during a specified day interval time for Symmetrix Device 0FC volume accessed through Symmetrix channel interface SA 13B from server esix49. Having these info make easy to understand if the workload is cache friendly or unfriendly and therefore we can easily estimate the workload over the backend.



Logical ...	ios per sec
0x0FE	4,56
0x100	3,47
0x0FF	3,42
0x0FD	3,40
0x0FC	3,35
0x101	3,24
0x103	3,24