Distributed Storage Systems

By Anil Nair, Cohort 4

Supervisor: Frank Bott
University of Wales, Aberystwyth, UK
5th September 2003

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the University of Wales.

DECLARATIONS

The content of this dissertation is the result of my own independent work and investigation except where otherwise stated. All sources are acknowledged by explicit references to the bibliography.

Signed: . . . . . (Anil Nair)

Date 5th September 2003

I declare that this work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: . . . . . (Anil Nair)

Date 5th September 2003

I hereby give my consent to the dissertation, if successful, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed: . . . . . (Anil Nair)

Date 5th September 2003

Acknowledgements

I am thankful to my supervisor Frank Bott, for his assistance and planning ideas for my project. I am also grateful to my colleagues in Singapore for constantly keeping me posted on critical dates updates from University professors regarding the dissertation. As I am currently employed in the US, meeting our professors in person became harder. I am thankful to the King County Library System in Washington for providing a wealth of information in the form of books and publications.

Finally I would like to thank my beloved wife for all her understanding and care for the time I had to spend aside for working this project.

Abstract

Developments in various areas of information technology particularly in the field of multimedia, have led users to demand both the capacity to store ever-increasing quantities of data and very high availability of that data. Distributed storage systems such as Network Attached Storage and Storage Access Networks address those needs. This dissertation discusses these technologies, their architecture and access methods. It discusses their implementation methods via hardware and software for improving availability of data and resilience of the systems. In conclusion it discusses the effectiveness of such technology, and addresses some of the issues for which they were designed.

Table of Contents

1. Introduction

Corporate data accumulation is increasing at a very alarming rate. Financial institutions add up data up to 20 Terabytes in 10 ~ 20 years while other companies dealing with multimedia storage can stack the same capacity in less than a year. The incredible growth in data intensive applications and the Internet has fuelled the growth of raw data storage. Data warehousing, online transaction processing, multimedia Internet & intranet browsing has doubled capacities in global 2000 companies and the growth is at the rate of five to eight times for e-commerce companies. The storage services revenue is expected to grow from $26 billion in year 2002 to up to $41 billion in year 2005[1].

1.1 Before the Advent of Distributed Storage Systems

Storage appliances in the past were directly attached to a general purpose Ethernet Network (Figure 1). Most of the storage was accessed via the server's main application. This model severely affected performance of data availability. Multiple requests from clients, server's internal processes, CPU utilization etc slowed down overall performance. The servers own applications and processes use to choke on the requests from the various clients during data access. The performance gets worse when a server needs to access data for a client located on another server, affecting overall network performance.

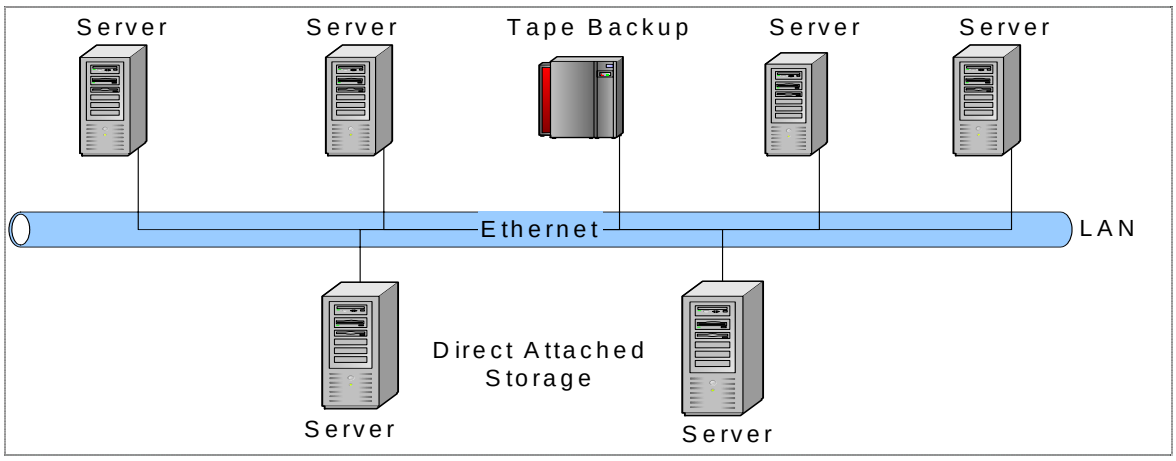

Figure1: Simple LAN Structure with direct attached storage

The process for adding more storage comprised of several painful steps. Firstly it had to be announced to affected parties that the servers wouldn't be available for certain duration of time. The work then gets carried over the weekend or during non-business hours, with data not being available for users during that time.

It was clear that storage equipment needed to be isolated from the servers.

[1] Gartner Dataquest, *Worldwide Storage Services Market, February 2002. Reference [20]*

1.2 The Need for Distributed Storage

Such server-centric storage solutions posed as a problem for today's non-stop pace of business, where any such downtime is unacceptable[2]. Many companies need the 24/7 uptimes. Today's Networks are becoming far busier than it used to be with desktops or clients downloading and sending large amounts data. They are no longer boxes, which compute data, but also communicate via email, streaming data via the Internet, making online telephone calls, video conferencing, and more and on some occasions many such applications at the same time. Such messaging and multimedia applications are demanding more and more bandwidth. Any application requiring data from another location needs to be highly available. Many businesses thrive on the technology of the Internet by providing services to consumers such as personal image studios, video on demand etc. Other groups such as in the medical field may need to hold sensitive and important patient information for many years due to country regulations. Such businesses and organizations heavily depend on Storage Servers for residing all this data. This data needs to be always available, resilient to many environmental and other factors that could result in potential data loss or downtime.

Studies have shown that storage servers are the greatest inhibitors of continuous data access. To help deal with this explosive growth in data, a data center approach is being adopted. Intelligent devices called Network Attached Storage appliances or NAS offer this. A NAS appliance is a networked product designed to de-couple servers from storage systems. The idea behind decoupling servers from storage systems is to improve data performance, reliability and scalability. It also facilitates the scaling of CPU utilization and storage capacity.

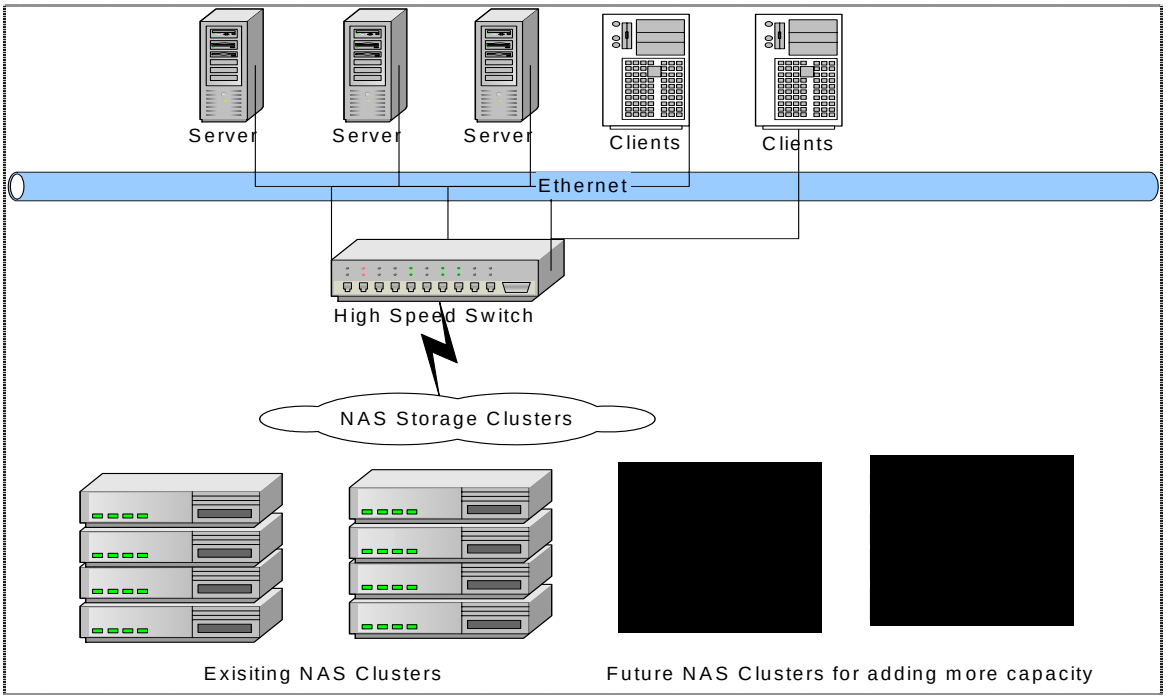

Figure2: NAS Infrastructure

[2] Byte & Switch Article, *Barry University email crash: Reference 10a*

The traditional LAN has storage systems within the same backbone as servers and workstations. NAS on the other hand has a dedicated storage network with its intelligence within a high-speed network and separated from the LAN environment, thus making additions in storage capacity much simpler without interrupting business operations. Figure 2 above shows a NAS infrastructure. A well-designed NAS can significantly reduce costs, installation time and overheads required as compared to direct attached storage appliances.

Newer developments in storage such as NAS is desired where additions in storage capacity does not require any downtime and without the need for server-server reconfigurations. NAS additions can be done when users are online and accessing data.

1.3 Objectives of this dissertation

The main objective of this survey is to provide a survey of the main distributed storage technologies and its architecture that is currently in the market. It aims at providing examples of how they are used and description of the reliability, performance and cost of the solutions available.

1.4 Methodology

The dissertation has been approached in a non-pathological way by providing concepts of distributed storage systems, followed by its technology. This dissertation is a compilation of all this from various sources, personal experience and knowledge. I am working in an industry which implements distributed storage appliances and hence conceptually it was appropriate to start a project around this topic. I had a good knowledge about its basic technology and implementations. And that is where I began. Later there was a need to understand its core technology, which I acquired with the help of various tech books and web sites. I have made attempts to layout this dissertation in such a way that the topics flow without having too many cross references, but at the same time explaining concepts along the way so that the reader doesn't have to scroll back and forth to understand a particular point. Also getting customer experiences by visiting people who use distributed storage systems helped understand the technology limitations. All this helped put the pieces together to make this whole project more complete and concise.

1.5 Structure of the Dissertation

In this dissertation I will discuss at length the distributed storage architecture and the various types of solutions available today. Chapter 2 discusses redundancy when storing data and explains the various solutions available that provide the different redundancy required. File allocations and data block handling is explained in Chapter 2 under Journaling (file systems). Data striping is described under Chapter 2. All data in most storage architectures available today is stored on hard drives using RAID technology, chapter 3 describes hard drive architectures and types, namely SCSI and ATA. Solutions such as NAS rely on TCP/IP and Ethernet for connectivity. Chapter 4 has a brief description of the various connectivity options available today. It also discusses fiber

channel technology currently deployed in some storage networks (SAN). Chapter 5 and 6 discusses networking protocols namely SNMP, SAMBA and NFS. Data Backup and recovery processes are explained in Chapter 7. I have included a summary for this dissertation in chapter 8 and finally an evaluation and critical appraisal under chapter 9.

2. Distributed Storage System Architecture

Distributed Storage Architectures falls under two types: Network Attached Storage (NAS) and Storage Access Networks (SAN). In this dissertation NAS has been picked for

2. 1 NAS Topology

NAS and SAN technologies are converging and resulting in solutions that have benefits of both. Both technologies also share a number of common attributes. Both provide optimal consolidation, centralized data storage, and efficient file access. Both allow user to share storage among a number of hosts, support multiple different operating systems at the same time, and separate storage from the application server. In addition, both can provide high data availability and can ensure integrity with redundant components and redundant array of independent disks (RAID)[3]. Following Figure 3 shows the hardware topology for a typical NAS solution.

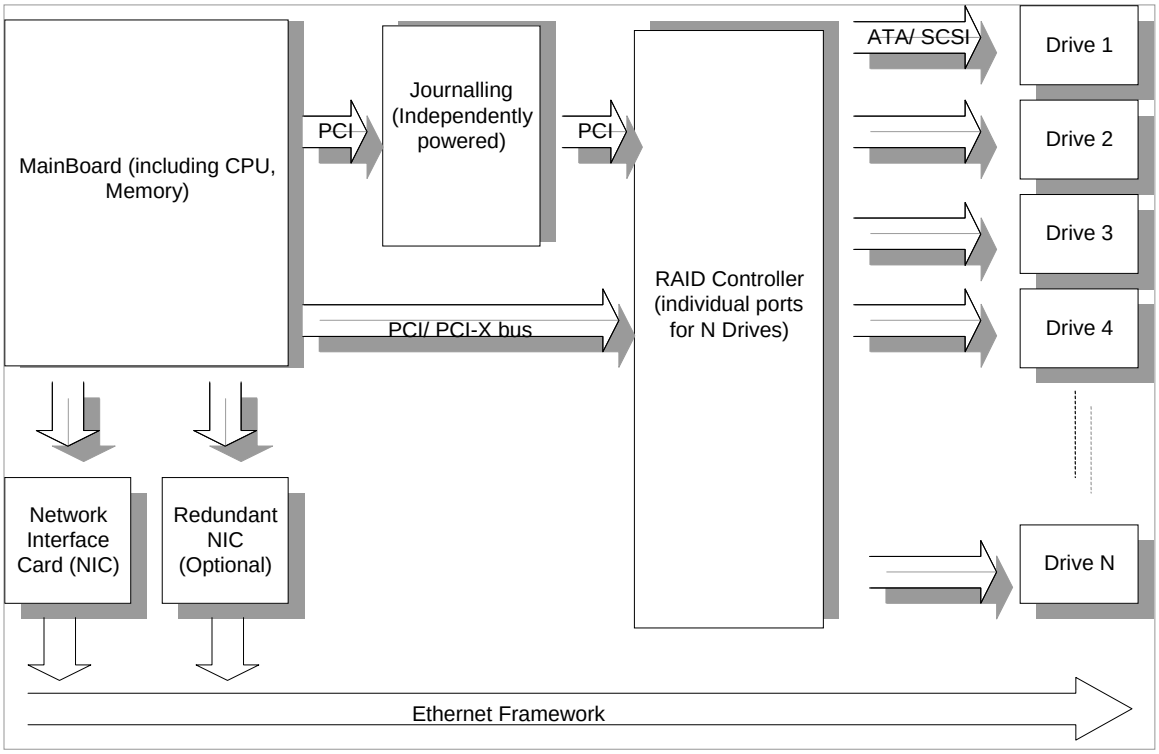

Figure 3: Hardware NAS Topology

2.2 NAS or SAN

Some may view NAS as competitive to SAN, when both can, in fact, work quite well in tandem. NAS and SAN represent two different storage technologies and they attach to a network in very different places. NAS is a defined product that sits between the application server and the file system. SAN is a defined architecture that sits between

[3] Convergence of NAS and SAN (NetApps) – *reference [16]*

your file system and the underlying physical storage. A SAN is its own network, connecting all storage and all servers. For these reasons, each lends itself to supporting the storage needs of different areas of a business.

Some notable differences need to be however pointed out.

- SAN solution uses fiber channel (section 9 of this dissertation), whereas NAS uses TCP/IP connections such as Ethernet, ATM[4]. SAN uses filer protocols whereas NAS uses NFS (Network File System, section 12).
- SAN has the file system resident on a separate appliance from the storage devices whereas NAS has the FS within the storage devices.

A typical NAS product is essentially a high end-computing product with a super large storage capacity. The main differences arise where the technology is tuned to allow for high data storage, data integrity, and scalability within many NAS systems, cache buffer and high-speed performance. A typical NAS product calls for a motherboard with a high speed microprocessor coupled with multi-port RAID cards, redundant network ports, journaling memories for buffer cache and storage disks connected as RAID or JBOD (just a bunch of disks) to the RAID card. If the disks are configured as JBOD then the redundancy capabilities are typically managed via software RAID.

2.3 Redundancy

Redundancy is key in storage systems and can be implemented in several areas in the hardware where failures are known to occur and inevitable at any time.

2.3.1 System Level Redundancy

For a system level, the redundancy can be incorporated on specific hardware that is known to cause issues, which may be externally induced. Some of these failures includes: unavailable Ethernet links (due to a bad switch), loss of power from the AC circuit, bad or failing disk drives etc. One can decipher which part in the system requires redundancy such that the system continues to perform. Following are some of the redundancy characteristics that can be considered for achieving this:

- Power Supplies can be made redundant (two or more power supplies hooked together within the system and powered on) such that they are load sharing and operating. In case one of the power supplies fails, then the other one takes over 100% and continues to deliver necessary power. Also failures can be external to the system, for example a single phase can go down, taking down the system. To prevent this each of the redundant power supplies can be plugged to different phase circuits.
- Dual or Quad Ethernet ports can be implemented rather than just a single Ethernet port. This would provide redundancy as far as accessing the system on the network. The system can be wired to different switches on the same network/ subnet. This will allow continuous access on the system in case one of the switches malfunctions.

[4] Teach Yourself TCP/IP (Timothy Parker). *Reference [1]*

- Disk drives are mostly mechanical devices and prone to failures at some point or the other. And storage appliances have to rely upon disk drives for data integrity and availability. Redundancy on disk drives is of utmost importance and can be implemented in several levels as explained in the next section.

2.3.2 Cluster level redundancy

In a cluster of several systems, failures can occur with one particular system. The redundancy can be implemented to the extent that the data can be available even if one entire system goes down. Mirroring the same data to other systems will allow this protection. Mirroring is set when the cluster is configured. Depending on the protection levels, mirroring can be done per system or on multiple systems. Mirroring however has overhead costs due to extra megabytes being written onto disks for the desired protection. Also when a single system fails this needs to be fixed immediately because if another failure occurs before the MTTR (Mean Time to Repair) then there is danger of losing data, as the file system would be in a completely unprotected state.

2.4 Journaling

The issues network administrators face is the inevitable cases of power loss to the system. This is minimized by the use of backup power supply etc, but loss of power is still a possibility at times and that results in the file system having to run a file system check as it powers back up and this will take hours and hours to complete. Not to mention data lost during any input/ output transactions that could have happened at the time with power loss occurred.

There are several static file systems but they do not guarantee that the data updates have been done to the disks safely. In the commercial arena static file systems cannot be trusted. This is where journaling file systems come in. It's far more superior to static FS and has even better performance.

2.4.1 How a File system works

First a few words on how a file system works. File systems stores data to hard drive by determining where each file's blocks, or pieces of the file content, should be stored and by maintaining a table of the locations of those blocks. The data structure specifying the location of a block, or a set of blocks, is called an *inode* (index node). The methods for allocating and retrieving file blocks determine the overall performance of reads and writes to disk and the reliability of the overall file system itself.

Static File systems like *ext2fs* (Linux)[5] maintain a structure of such *inodes*, which points to different data blocks it contains. Every *inode* has a unique number that will distinguish itself from the rest. It also contains information regarding the files permissions and ownership information called *metadata*, list of blocks, indirect blocks etc. Following example (Table 1) shows the *inode* information for a file called sample.file

[5] Running Linux (Matt Welsh). *Reference [6]*

Sample.file *inode* number	1202
Metadata	Author: Anil Nair, Title: Sample file, Share: Restricted, Modified Date: 01/20/2003, Modified Time: 09:34:02 …
Data Blocks	1110, 1111, 1112, 1113, 1145
Indirect Blocks	
Etc..	

Table 1: *inode* information for sample.file

The file sample.file has data blocks at 1110, 1111, 1112, 1113, and 1145. These are essentially locations on the disk. Note that after 1113 the next block for the file is at 1145. This is due to the fact that the disk may be fragmented and there is data residing already between blocks 1113 and 1145. There are no indirect blocks in this example but this is possible if the file had been large in size. These indirect blocks can then reference to double indirect blocks and so on. So in order to read the file sample.file, the disk has to read off block on 1110, 1111, 1112, and 1113 and then move the head all the way over to 1145 to retrieve the entire file.

If the file needs modification by the user on the blocks located at 1112 and 1113, then the data is read off these two blocks and rewritten with the updates into the same blocks. If the file has additions to itself, say two more blocks worth of data, then these two free blocks (say 1150 and 1151) will be found on the disk and written onto, thereby increasing the number of blocks the file carries. This will also change the *metadata*. The *inode* info for sample.file will now look like below:

Sample.file *inode* number	1202
Metadata	Author: Anil Nair, Title: Sample file, Share: Restricted, Modified Date: 01/20/2003, Modified Time: 10:34:02 …
Data Blocks	1110, 1111, 1112, 1113, 1145, 1150, 1151
Indirect Blocks	
Etc..	

Table 2: Updated *inode* information for Sample.file

Now consider a vast amount of change that is being implemented to the data structure. For example, if one has to update a directory entry and is in the middle of updating 20 file entries on the 4th block within a giant directory. Now during this update if there is a power outage then the write does not get completed resulting in a corrupted 4th block.

When the system does get rebooted then the file system runs a file system check (utility is called *fsck* for Unix, *scandisk* in Windows), which runs and looks for consistency in the file system and references blocks to the right files. It would finally then come to the corrupted block and attempt to repair it. But in many cases it would not be able to, leaving the block into a section onto the disk called the *lost+ found* associated with each file system. These blocks that are located in the lost + found section are actually in use but there is no way of knowing where they are referenced from. Running fsck will take a long time as users are waiting for the systems to come up. If the system has gigabytes

of data then file system check can take several minutes to complete. Many gigabytes can take the up to 20 minutes to ½ hour.

Larger systems with terabytes of data, such as in distributed storage system; this poses an unacceptable delay in data availability and downtime.

Journaling file systems or logging file systems takes care of this issue. In a journaling file system a snapshot of the changes being made to a file in a section in the directory such as *log.*

It keeps track of the *metadata* of the file being changed and also block information such as modification and addition of new blocks to the file. This information is retained in the *log*. Once the file changes are completed and to be saved, the file system will then read this *log* information for the file and make changes to the file and directory *inodes*. This process is called *committing* the file. Whenever a file gets accessed the *log* is consulted to ensure that it does not have any uncommitted states. Every once in a while the file system with review the log and trim it down in order to reduce the access time. Every file access, which results in a modification to it will have, an *intent-to-change* marked in the log. This is reviewed against a *commit* state to ensure whether the file was successfully saved. If we take the previous example where a file (*sample.file*) had some blocks modified and some blocks added, the *log* would keep track of these changes in metadata and directory *inodes* and mark an *intent-to-change'* for the transaction. Now if there is a power loss event, this information still exists in the *log*. Thus when the system does recover and come back this *log* or journal is replayed. The *intent-to-change* for sample.file is reviewed. The following (Table 3) will show the logstatus of sample.file after recovering from a crash:

Inode: 1202	Intent-to-change
Block 1110	Data & Change information
Block 1111	Data & Change information
Block 1112	Data & Change information
Block 1113	Data & Change information
Block 1145	Data & Change information
Block 1150	Data & Change information
Block 1151	Data & Change information
Inode: 1202	Modify Block 1113 & 1145
Inode: 1202	Add new Blocks: 1150, 1151
Inode: 1202	Update data block list: 1110, 1111, 1112, 1113, 1145, 1150, 1151
Inode: 1202	Access Date: 01/20/2003, Access Time: 10:40:02
Inode: 1202	Modified Date: 01/20/2003, Modified Time: 10:45:02
Inode: 1202	Not-committed

Table 3: Journal Replay for Sample.file

The *log* will also know that the file was not committed. The *log* will then resolve such *in-doubt transactions* and make appropriate changes to the file such as modification of

blocks and addition of new blocks and *commit* them to disk, thereby resulting in no data loss.

Journaling file systems can come on line on line instantly as compared to static file systems. Any crash information can be fixed by *replaying journal* and changes to all existing file structures are committed by reviewing the *journal log*. No data loss occurs except for changes that were being made at the time when the system went down. Such file systems however will come with a cost, because journal transactions requires frequent writes to the disk itself to maintain log information. It needs to write to disk to append changes in file structures and then later replay that information before committing the file changes to disk. However file systems can be designed to make the commit entries into its *log* during the time it is idle. Also logs are usually appended to itself thereby making replay easier than the actual modification.

Journal files systems ensure that during transaction events when no transaction is left in an inconsistent state in the event of a power loss. Journaling file systems that use the disks to create logs will take a huge performance hit due the fact that the journal replay process requires access to drives that contains the log files as well as actual data though they may be residing on different partitions on the drives. This can be overcome by using separate journaling hardware such as solid-state memory with adequate capacity for having a journal or log file. Solid-state memories though are expensive, can improve performance due to its fast access times compared to the much slower hard drives. Journal replays can be done at a much faster rate, and allowing it to only commit affected files enhances disk drive performance. Solid-state memory however requires continuous power thereby having to rely upon fast separate power considerations such as rechargeable batteries etc.

3. RAID[6]

RAID stands for Redundant Array of Independent Disks and fundamentally stands for combining several hard disks into one logical storage unit. The mechanism to do this is called *striping*. The striping can be done such that they are partitioned between drives into sizes as small as the sector size (512 bytes) or up to several megabytes. This method allows for maximum performance. In the event there are input/ output requests to certain files several disks can be put to work to get the file bits for faster availability. In I/O intensive environments the data can be made available in one single stripe thus making the record potentially available in that single stripe. Rich media such as video, audio files, which utilize long record sizes, can be stored in smaller stripes.
When striping in small sizes in a RAID, one needs to make sure that the speeds of the drives where the stripes are being made are identical. If it is not then the slowest spindle speed will determine actual file access time. Hence most systems need to be implementing identical drives for storage.

3.1 Various RAID Levels

There are several ways RAID can be implemented in a storage system. Following list most of the types currently implemented.

3.1.1 RAID0

Striping of data as mentioned above is performed in this type of RAID. However there is no redundancy on data. The data is only broken down into smaller blocks and striped across the various disks in the array. This increases the throughput on the file whenever it is accessed.

3.1.2 RAID1

This level allows data to be written to two or more drives. This allows redundancy of the file and is called data *mirroring*. In any event a drive fails then data is still not lost. Though this is a great advantage the cost per megabyte increases.

3.1.3 RAID2

This level incorporates Hamming Code ECC (Error Code Correction) when written to disks. Upon a read of these files the ECC code gets verified and is corrected if there are any errors. Most disks these days SCSI[7] or ATA have in-built ECC correction hence this RAID level is rarely used in configurations.

3.1.4 RAID3

Data is striped in this RAID at the byte level on multiple disks and the parity information for these are written onto another parity disks, which gets verified upon reads.

---

[6] Managing RAID (Derek Vadala). *Reference [7]*
[7] The Book of SCSI: I/O for the Millennium (Gary Field). *Reference [4]*

3.1.5 RAID4

This is quite similar to RAID level 3 except that the data is striped at block levels. Parity information is still stored in the parity disk. This RAID however has a pretty poor write average transfer rate.

3.1.6 RAID5

This is similar to RAID4 except that parity information is written on individual disks rather that a parity disk. This removes the bottleneck seen to access the parity disk in the cases above.

3.1.7 RAID6

Similar to RAID5 except that it requires a second independent level of parity in the same disk. This is useful for mission critical operations. However it takes a huge hit with write performance.

3.1.8 RAID7

This also has a dedicated parity drive. There is an embedded Real Time operating system controlling the communications channel. All I/O transactions are asynchronously and independently controlled and cached including host internal transfers. Overall write performance increases from 25% to 90%.

3.1.9 RAID10

This level has mirroring and striping combined into one. This is useful for high fault tolerance and high performance. There are huge overheads for this and is very expensive (cost per megabyte).

3.1.9 RAID 53

This type is a striped array like RAID0 whose segments are like RAID3. It has the same fault tolerance as RAID3.

3.1.10 RAID 0+1

This is implemented as a mirrored array whose segments are striped like in RAID0. This solution is very high performing, but not well for high reliability. A single drive failure will turn the array into RAID0 before data is regenerated.

3.2 Hardware RAID

Hardware RAID controllers today manage the RAID subsystem independently of the host and present the storage as a virtual disk. The host need not worry about the RAID subsystem. Most controllers provide RAID solutions using hardware. There are solutions where the RAID is external. The RAID subsystem is connected via a SCSI (or ATA)

(section 14) controller and appears to the host as a single disk. The connections are made using fiber channel. However single SCSI channels used creates a bottleneck.

3.3 Software RAID

There are many controllers that provide multiple disk drives over the SCSI or the ATA channels but do not have any RAID capability. They provide such disks to the host as just a bunch of disks (JBOD). The host in turn needs to have drivers to support RAID and its various levels. This poses issues, as every OS needs to have its own special drivers written to support it.

3.4 Software versus Hardware RAID

Software based arrays occupy host system memory, consume CPU cycles and are operating system dependent. By contending with other applications that are running concurrently for host CPU cycles and memory, software-based arrays degrade overall server performance. Also, unlike hardware-based arrays, the performance of a software-based array is directly dependent on server CPU performance and load.
Hardware Arrays are highly fault tolerant. They are OS independent. They have the array algorithm in-built into the firmware hence can boot into any disk without the need for software. This is the primary issue with software RAID implementation, when a boot drive gets faulty, then the entire device may be unusable. The workaround is to have fault tolerant solutions developed in software RAID where the there can multiple boot drives, such that whenever one of the boot disks fails, the system should failover to the next available good disk, and then later on mark the original drive bad for replacement. Mirrored data off this disk will then be re-striped onto the good disks to achieve the necessary protection. As soon as the new disk is installed the data is then written back to the new drive and data is balanced across the drives.

3.5 Data Striping

The following explains the various striping process implementations:

3.5.1 Unprotected Files or 1x Mirroring

Each *inode* has a variable that specifies the width the *inode* should have. The width is the number of drives that will participate in storing the data blocks of the file. This width is sometimes called a "stripe count." These files can be created with a width of ‘n’. If there is less than ‘n’ drives in an array, the width is set to that number of devices in the array.

Data blocks are distributed evenly among all participating devices. For example, the following map in Figure 4 might describe a file that has a width of 8 devices (denoted in the top row).

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

16	17	18	19	20	21	22	23

Figure 4: Data Blocks of a file distributed between 8 devices

The numbers 0 - 23 are logical block numbers and each number represents a data block of that file. Each drive is represented by one column in the map as shown within the square brackets. Data blocks #0, #8 and #16 all reside on the first drive [1] in this example. As we see here there is no protection for this file at all. If drive [1] fails we lose data blocks for that files and thus result in file corruption.

3.5.2 Mirroring

Mirrored files are distributed or striped similar to unprotected files. Each mirrored copy of the data block is striped exactly like the unprotected files described above. Mirrored copies of the data are offset from each other so that the corresponding data blocks in each separate mirror are all on different devices.

[1]	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**[8]**
0[0]	1[0]	2[0]	3[0]	4[0]	5[0]	6[0]	7[0]
8[0]	9[0]	10[0]	11[0]	12[0]	13[0]	14[0]	15[0]
16[0]	17[0]	18[0]	19[0]	20[0]	21[0]	22[0]	23[0]
7[1]	0[1]	1[1]	2[1]	3[1]	4[1]	5[1]	6[1]
15[1]	8[1]	9[1]	10[1]	11[1]	12[1]	13[1]	14[1]
23[1]	16[1]	17[2]	18[1]	19[1]	20[1]	21[1]	22[1]
6[2]	7[2]	0[2]	1[2]	2[2]	3[2]	4[2]	5[2]
14[2]	15[2]	8[2]	9[2]	10[2]	11[2]	12[2]	13[2]
22[2]	23[2]	16[2]	17[2]	18[2]	19[2]	20[2]	21[2]

Figure 5: Block distribution with mirroring

The default width of a mirrored file is whatever the mirror count is set to. If the mirror count is less than 8, however, the default width is 8.

For example (shown in figure 5), lets assume that a file needs to be mirrored 3 times and has a width of 8 (drives) might be described by the following map. The first number represents the logical block number of a data block, and the number in square brackets is the mirror number of the data block. The topmost row denotes the drive number.

In the above example Data blocks #0, #8 and #16 are mirrored within Drives 1, Drive 2 and Drive3. This raises the level of protection for the file three times at the expense of drive capacity.

3.5.3 Files with Parity Protection

Parity protected files are determined by two factors: the parity count and the width. The parity count is the number of data blocks that are XOR'ed together to create each parity block. A file that has 4+1 protection policy has a parity count of 4. The default width for a parity-protected file is the parity count plus one.

The data blocks together with the parity block that protects them are collectively called a *group* in the source.

Parity blocks are distributed evenly among all participating devices. Usually parity blocks are the last block in the group. Sometimes they are shifted to a different position in the group in order to create a more even distribution.

For example, the following map might describe a file that has a parity count of 3 and a width of 6, providing the 4+1 protection. The numbers by themselves are logical block numbers of data blocks. The numbers next to a letter P are the logical parity block numbers of parity blocks.

[1]	[2]	[3]	[4]	[5]	[6]
0	1	2	P0	3	4
5	P1	6	7	8	P2
9	10	P3	11	12	13
P4	14	15	16	P5	17

Figure 6: Files protected with parity

In this example, data blocks 0 - 2 are protected with parity P0. The next group is data blocks 3 - 5 and with parity P1. The third group is data blocks 6 - 8 and parity P2. The parity block is at the end of each of these groups. In the next three groups, the parity block has been shifted, so that it is placed second from the end of each group. This is because if the parity block were at the end of those groups, there would be more than one parity block on some devices and no parity blocks on other devices.

The number of times a parity block gets shifted can be kept track by storing this into a variable (such as `totalShift`). At the start of each file `totalshift` will be 0 and the parity block is at the end of each group. After a number of datablocks are written the parity block gets shifted and `totalShift` gets incremented. The variable `totalShift` can be calculated by:

```
totalShift = LCM(parity_count +1, width)
```

For this example, the shift width is the least common multiple of 4 and 6, which are 12.

```
totalShift = 2 x 2 x 3 = 12
```

In other words, after every 12 blocks the parity block position is shifted to the left relative to the other blocks in its group.

After one parity block has been placed on each device, then the cycle starts over with the parity blocks appearing at the end of each group again. The number of blocks in each cycle (e.g. `cycleWidth`) is calculated by:

```
cycleWidth = (parity_count + 1) * width
```

The height of the cycle (the number of rows in the diagram) will be always equal to `(parity_count + 1)`.

3.5.4 Clustering

To improve throughput, data blocks and parity blocks can be clustered on each device in the array. A typical cluster may have 16 contiguous blocks of say 8K each.

Each parity block will actually represent 16 parity blocks in a contiguous cluster.

For example, the unprotected file from the first example would actually be written to disk like the following map in figure 7 shows. Each group of numbers shows the range of logical block numbers in the cluster.

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
128-143	144-159	160-175	176-191	192-207	208-223	224-239	240-255
256-271	272-287	288-303	304-319	320-335	336-351	352-367	368-383

Figure 7: Clustering

For all other maps above, the logical block number in the diagram is similarly representative of 16 blocks.

3.6 Hard Disk Drives

Hard Disks were invented in the 1950s with a few megabytes of storage. Since then Hard disks have grown in capacity as well as speed. Drive capacities currently reach up to 250 Gigabytes with 300+ GB and above already in the pipeline[8]. Plus the interface can be either IDE or SCSI. The basic construction has remained the same over the years with data storage available over a hard *platter* storing magnetic medium. The data is written to or read off platters by the means of read/ write head mounted on an arm, while the platter rotate. The aerial density in these disks has changed with manufacturers packing more and more capacities into the same platter space.

The platter comprises of sectors and tracks. Tracks are concentric paths on the platter filled with sectors and each of these sectors can be 256 or 512 byte. These sectors are usually grouped together at the operating system or drive level.

Disk platters vary from 10G ~ 80G. The precision required to read high capacity platters is even higher as the head has to write and read from sectors that are even smaller. Each track has a certain number of reserve sectors that are kept aside to replace any bad sector. Figure 6 above shows the sequence in which the drive processes a block of data for a write or read event.

Hard Drives are extremely sensitive devices. Any physical handling issues can seriously impair the performance of a drive, where the read/write head may ending up slamming

[8] Maxtor & Seagate Roadmaps. *Reference [12 & 13]*

against a platter chipping the magnetic surface leading to physical sector damage, or even arm alignment variances affecting write performance. If there was a bad write due to physical sector damage or due to a write exceeding its allocated sector (off-track errors), this causes an ECC (Error Correction Code) error. An ECC is computed and used however only during a read operation. The Hard Drive has in-built algorithm to recover the bad write block by using this ECC code.

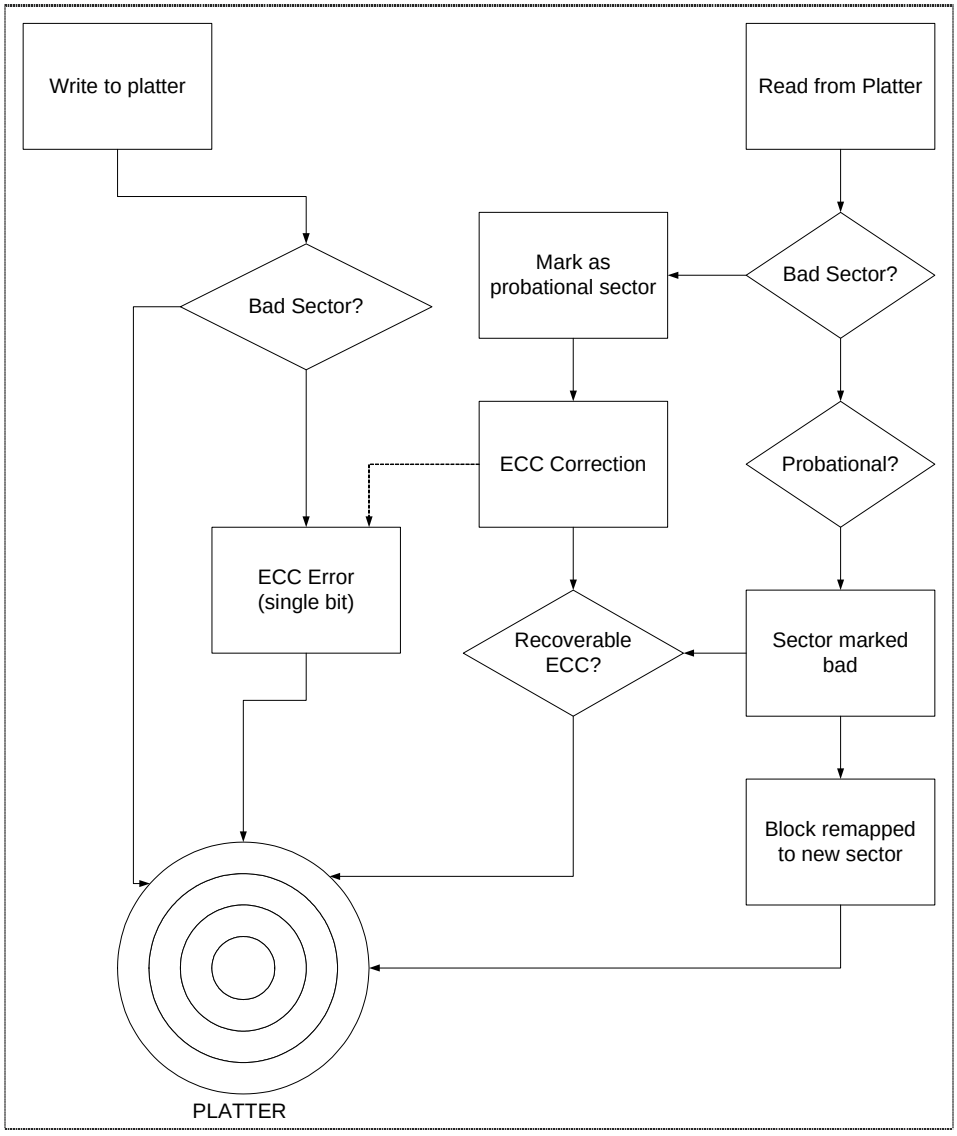

Figure 8: Read-Write event on a hard drive

3.6.1 Hard Drive Types

Today distributed storage vendors have a choice of using either SCSI or IDE hard drives for their storage solutions.

3.6.1.1 SCSI Hard Drives[9]

SCSI stands for Small Computer Systems Interface. The devices on the SCSI bus talks to the computer through one of the devices connected to the SCSI bus. That device is called the controller and has an interface to one of the data buses of the computer. Personal computers these days implement the SCSI controller communications to the PCI bus either as an on-board solution on motherboards or as a separate PCI card in a PCI slot.

One can drive SCSI drives using other protocols available today such as FireWire or IEEE 1394 or USB 2.0.

Each SCSI device can have multiple *logical sub-units*. All devices have the ability to release the bus after being asked to do time consuming operations not requiring the availability of the bus and leave it free for other devices to use for transferring data or receiving commands. The American National Standards Institute or ANSI defines today's SCSI standards, which describe the characteristics and capabilities of the interface. There are currently two main such standards, SCSI - 1 and SCSI - 2.

SCSI-1 is the original SCSI standard that describes the cable requirements, command sets and different transfer modes. Using an 8-bit wide bus where data transfers are done using a 5 MHz clock hence resulting in a 5 MB/s peak transfer rate.

At most 8 devices can be connected to the bus as a direct consequence of the bus width as each device is addressed using one of the 8 data lines. Most devices used on this bus are hard drives, as the command set doesn't explicitly support other media in the command set.

SCSI-2 -This standard was approved by ANSI in 1990. Transfer rates improvements was accomplished by doing two things:

- Increasing the data-clocking rate to 10 MHz.
- Increasing the bus width to 16 bits from the original 8 bits allowed doubling the transfer rate.

A side effect of this is that a wide SCSI bus can use 16 devices. Command set improvements were made and this made it possible to connect devices to the SCSI bus that previously required proprietary controllers like CD-ROMs. Cable design & specifications had to change to support the higher data clock. Command queuing is a benefit of SCSI and allows multiple outstanding requests between devices on the bus. A maximum of 256 commands on each Logical Unit Number of a SCSI device can be queued.

SCSI-3 - In order not to stop development it was decided that future development of SCSI should be divided into different layers and command sets. This has made it possible to develop physical transports using new technology like fibre channel for

[9] Book of SCSI (Gary Field). *Based on reference [4]*

massive increases in data transfer rate while at the same time keep and develop the older technologies like the parallel interface.

The most common SCSI interface today is the parallel interface and the improvements in SCSI-3 made compared to SCSI-2 are again higher data clocking speed and better cabling to handle the higher speed.

3.6.1.2 ATA Hard Drives

ATA or Advanced Technology Attachment and IDE or Integrated Device Electronics are the same, a disk drive implementation that integrates the controller on the disk drive itself. This was directly connected to the I/O bus of the first PC - the IBM AT. As a consequence, the bus width is still 16 bits on all implementations.

3.6.1.3 Data transfer modes for ATA Drives

PIO-Modes

PIO stands for Programmed Input/Output and was the standard way of using ATA devices but it's getting obsolete and making way for DMA and now UDMA modes. There are five different PIO modes, each have different transfer rates. The higher the mode numbers the higher the transfer rate. All PIO modes use the CPU to transfer data, which makes this method unsuitable for multitasking environments.

DMA modes

DMA stands for Direct Memory Access and is the term used when a peripheral device transfer data directly to or from memory, without the use of the CPU. Today DMA is the only feasible way to transfer data from hard drive to memory as most operating systems today use multitasking and can better use the CPU for other tasks. The first DMA modes were not adopted by the popular operating systems of the time, but when Ultra DMA mode entered the scene it quickly became commonplace. The main difference between Ultra DMA and the older single word and multiword transfers is that Ultra DMA mode clocks the data twice per clock cycle thereby doubling the bandwidth.

3.6.1.4 ATA Standards[10]

ATA-1
To eliminate some major compatibility problems with the early ATA/IDE drives the ATA-1 specification was defined as an ANSI standard in 1994. Previously the most common problem showed up when drives of different manufacturers were placed as master and slave on the same channel.
The original ATA/IDE standard defines the following features and transfer modes:

[10] ATA Specifications. *General reference [22]*

It supports one or two hard drives on the same bus. One is configured as master and the other as slave. These are usually selected by the means of a jumper physically present on the drive.

- PIO modes: 0, 1 and 2.
- DMA modes: 0, 1 and 2 and multiword DMA mode 0.

ATA-2
The ATA-1 standard defined what the interface was capable of a decade ago at its original inception and there was a need for faster transfer rates and enhanced features. The ATA-2 standard was defined as an ANSI standard that is backward compatible with the older ATA-1.

- Faster PIO modes: 3 and 4.
- Faster DMA modes: 1 and 2

ATA-3
In 1997 this ANSI standard was defined and can be viewed as a minor revision to ATA-2 and includes improves the reliability of the faster transfer modes introduced with ATA-2. Also added was the open standard for monitoring disk drive health called SMART.

ATA/ATAPI-4
This revision adds some significant and long-awaited features:

ATAPI (*AT Attachment Packet Interface*) are for devices that require commands not available in the standard ATA standard like CD-ROMs and CD-R.

Ultra DMA data transfer protocol, a.k.a. Ultra ATA, which clocks data twice per clock cycle by using both the negative and positive transition. It defines an 80-conductor cable to be used for Ultra ATA devices, which reduces noise and crosstalk across the data lines. This is not mandatory however.

ATA/ATAPI-5
Adds Ultra DMA mode 4 or Ultra DMA/66, the 80 conductor cable (ground wire alongside every data line) is now mandatory to maintain signal integrity.

ATA/ATAPI-6
This adds Ultra DMA mode 5 or Ultra DMA/100. This breaks the 137Gbyte capacity barrier. With hard drive capacities increasing in aerial densities (up to 250G available today) this was a big requirement. Discussions are being held concerning noise reduction, which can be found in, drives shipping today as well.

ATA/UDMA 6
This increases the speed to Ultra DMA/133 MHz and is the latest implementation in the drives being shipped currently.

3.6.1.5 Cabling for ATA

It's easy to describe the different cable types used by the ATA interface today because there really is only one standard. And that is a 40/80 PIN flat cable with 3 IDC

connectors. We can attach up to two units on the cable, one master and one slave. The 80-pin cable is for use with Ultra DMA devices but the 40-pin cable can be used with newer Ultra DMA devices but no faster modes than Ultra DMA/33 are available in that case. In later years new cables have emerged and that is a 44 pin flat cable, which is mostly used for 2.5" internal hard, drives. The 4 extra pins are used for supplying power to the drive. As far as we know there does not exist any external IDE cables.

3.6.2 Future for ATA

3.6.2.1 Serial ATA

Based on [18]. The next generation ATA drives are going to be via a Serial interface. The serial ATA 1.0 (SATA) group was established in Feb 2000 to specify the needs for desktop PCs, later in Feb 2002 SATA II working group was developed to further address the needs for servers and network storage.

Figure 9: Parallel & Serial ATA Cable[11]

Performance increases over the parallel interface up to 150 MB/sec and the speeds are proposed to scale up to 600 MB/sec in the next few years. A major improvement of SATA over the parallel interface is the cabling. The figure above shows the reductions in cable form factor.

Serial ATA offers consumers a new level of interface scalable performance, flexibility, and cost efficiency. Industry leaders designed Serial ATA with customer convenience in mind by ensuring 100% software compatibility, flexible thin cables, hot plug connectors, and improved data reliability and protection.

The major improvement that we see with SATA implementation is the reduction of cable sizes to connect the drives. This immensely improves cooling within the device by allowing more airflow. The other advantage of SATA over parallel is the hot pluggable functionality. This puts the SATA technology up against the SCSI devices. Hot pluggable drives are almost a necessity in today's storage environment. The highest failing component for a storage appliance is the hard drive. Especially with 100% duty cycle[12] in the cases of distributed storage, this failure rates are even more. Hot plug functionality is a must. Parallel ATA technology does not support 'true' *hot swap*. The

[11] Serial ATA Committee (http://www.serialata.org/about/index.shtml)
[12] Powered on 24 hrs a day with i/o to the drive

drive does not do that anyway. It however can be done on the controller level. If a drive is going bad, system calls can be sent to the controller or RAID card to turn off i/o to a specific port (connected to the bad drive). This will stop access to the drives from any user. Another command needs to be sent to spin down the drives, after which the one can pull the drive out for replacement. If a drive is hot-plugged before its port is turned off and/or the drive is spun down then it results in either severe drive or controller damage.
On the other hand a SATA drive is designed to be hot pluggable. A user can plug a drive in and out when the system is active without resulting in any hardware issues. Even with this functionality there is a need for the application layer to be notified if a drive is going to be removed. If a drive is going 'sick' due to ECC errors, or high number of bad sectors, at the application layer, transactions needs to be terminated to that drive before it is pulled out. This is critical because once a journal (as discussed above) has been *committed* to disk; the data is destined to go to that particular disk. And if we pull that drive out at that time (SATA or Parallel ATA drives), it would result in data loss as the journal cannot find that drive.

3.6.3 SCSI vs. ATA

To make a fair comparison between modern SCSI (SCSI-3) and ATA (ATA/ATAPI-6) one has to look at two different scenarios: Single device and multiple device environments.

3.6.3.1 Single device

This scenario is common in desktop computers where we connect a single device to a single adapter and perform data transfers. There is practically no difference between the two interfaces, this holds for bandwidth as well as resource usage (CPU) as both interfaces use the most efficient way to transfer data, namely DMA. This means that there is no point in purchasing a generally speaking more expensive SCSI based system when the cheaper ATA interface would do an equally good job.

3.6.3.2 Multi device

This is where a distributed storage application comes in where we connect multiple devices to one or more interface adapters.

This is where SCSI has major advantages compared to ATA:

Connectivity: The ATA interface can only address two devices while SCSI can address eight devices (Narrow SCSI), 16 devices (Wide SCSI), 32 (Very Wide SCSI) or 126 (FireWire). There are also many peripherals available to SCSI only and not ATA.

Bandwidth: The demand for high transfer rates in servers cannot be met using current ATA interfaces based on the two devices per adapter limit and even if it could carry more devices there simply isn't enough bandwidth and flexibility available for serious server application.

Efficiency: The ATA devices lack the intelligence to perform command queuing as well as their SCSI counterparts, which can queue up to 256 commands per logical unit. SCSI hard disk drives aimed at the extreme performance server market have had a lot of research and development time on optimizing seek patterns and rescheduling commands to minimize seek times and maximize throughput. This may not be evident by looking at desktop benchmarks but under heavy server loads, this is evident.

Also, SCSI hard disk drives generally tend to be designed to work well in RAID-systems where I/O load is spread across multiple drives.

Dependability: Most high-end SCSI hard drives are quite expensive but there are good reasons for it. They can sustain higher temperatures and stay mechanically functional despite the expansion of the metal parts with temperature and generally have better build quality. The net result is that they are the natural choice for enterprise server applications. Connectors suitable for *hot-swapping* drives in RAID-systems is something only SCSI boasts, and helps maintaining large disk arrays where down-time is unacceptable.

4. Connectivity to Storage Networks

Some of the differences are pointed out in section 2.0 regarding NAS and SAN. Here we shall discuss one of the primary differences between the two with regards to its connectivity to the outside world.

4.1 Ethernet

A Network attached storage appliance, as the name implies, is connected directly to the network and clients access data directly without going via an application server. The *Ethernet* serves as a backbone for most NAS appliances. The IEEE 802.3 standard defines what is commonly known as the CSMA/CD protocol[13]. The term *Ethernet* refers to such a family of local-area network products covered by the IEEE. Three data rates are currently defined for operation over optical fiber and twisted-pair cables:

- 10 Mbps – 10Base-T Ethernet
- 100 Mbps - Fast Ethernet
- 1000 Mbps - Gigabit Ethernet

IEEE has also completed work on the 10 Gbps, which is the IEEE 802.3ae – an addendum to the IEEE 802.3 standards. Over the years Ethernet has survived as the primary network connection protocol covering 85% of the world network connections for PC or LANs. Ethernet LANs consists of network nodes and interconnecting media. Network Nodes falls into the two following categories:

Data Terminal Equipment (DTE): These are devices that act as the source or destination for the data frames such as PCs, Workstation, print servers, storage appliances etc.

Data Communication Equipment (DCE): These are devices that act as carriers for such frame and fall in between the DTEs. Examples of DCEs are switches, repeaters, routers, hubs, network interface cards or modems.

4.1.1 Network Topologies[14]

Point to Point

A simple network can be defined as a connection between a DCE and a DTE. The link between the two is the network connection.

Coaxial Bus Topology

Several DCE are connected to an underlying Ethernet fiber in series.

Star Connected Topology

[13] IEEE Ethernet Task Force, *reference [11]*
[14] Introduction to Data Communications & Networking. (Berhouz Forouzan). *Reference [8]*

Several DTE are connected to DCEs in a star fashion. This framework is in use extensively.

Ethernet's logical representation to the OSI Reference Model:

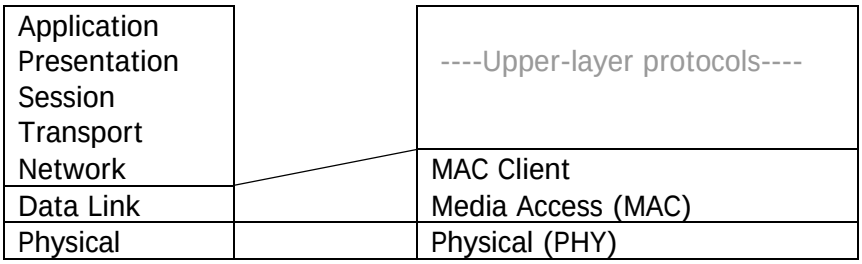

Figure 10: Ethernet & the OSI Model

The MAC sub layer can be a Logical Link Control (LLC) if the device is a DTE. This provides an interface between the Ethernet MAC and the upper layer of the protocol stack on the receiving end. The MAC sub layer can be a bridge entity if the device is a DCE providing an interface between LAN-LAN devices, which uses similar or different protocols.

Ethernet devices implement only the bottom 2 layers of the OSI layers; hence they are typically implemented as network interfaces (NIC) on a system motherboard. Many motherboards even provide multiple NIC in the system. This is essentially critical in storage appliances due to redundancy. If 2 or more NICs were present then the most reliable method would be to have each port connected to different switches on the same network. This eliminates the single point of failure for the storage device if one switch malfunctions, the system would still be accessible via the redundant NIC port.

As mentioned above with NAS we directly access data via a LAN or a WAN, but with a SAN we access it across the storage area network, thereby reducing the bandwidth load on the LAN or WAN. SAN uses a dedicated fiber channel network for data access.

4.2 Fiber Channel[15]

In distributed storage systems such as SAN Networks, there has been a growing need in the recent years to have extremely fast data links. High performance computers have become the focus of much attention in the data communications industry. Performance improvements have spawned increasingly data-intensive and high-speed networking applications, such as streaming multimedia and scientific visualization. However, the existing network interconnects between computers and I/O devices are unable to run at the speeds needed.

Fiber Channel serves as a backbone for SAN Networks. The intention of the Fiber Channel (FC) is to develop practical, inexpensive, yet expendable means of quickly transferring data between workstations, desktop computers and storage appliances (and

[15] Fiber Channel for SAN (Alan Benner*). Reference [5]*

even other peripherals). Fiber Channel is the general name of an integrated set of standards being developed by the American National Standards Institute (ANSI).

There are two basic types of data communication between processors and between processors and peripherals: channels and networks. A channel provides a direct or switched point-to-point connection between the communicating devices. A channel is typically hardware-intensive and transports data at the high speed with low overhead. In contrast, a Network is an aggregation of distributed storage nodes with it's own protocol that supports interaction among these nodes. A network has relatively high overhead since it is software-intensive, and consequently slower than a channel. Networks can handle a more extensive range of tasks than channels as they operate in an environment of unanticipated connections, while channels operate amongst only a few devices with predefined addresses. Fiber Channel attempts to combine the better of these two methods of communication into a new I/O interface that meets the needs of channel users and also networks users.

Fiber Channel actually represents neither a channel nor a real network topology. It allows for an active intelligent interconnection scheme, called a Fabric, to connect devices. All a Fiber channel port has to do is to manage a simple point-to-point connection between itself and the Fabric.

Fiber channel[16] is a high performance link that supports higher-level protocols such as SCSCI (for drives used in SAN networks). The Fiber Channel standard addresses the need for very fast transfers of large amounts of information. The fast (up to 1 Gbit/s) technology can be converted for Local Area Network technology by adding a switch specified in the Fiber Channel standard, that handles multipoint addressing. There is a perspective as an I/O technology and a Local Area Network technology as well. Another advantage of Fiber Channel is, that it gives users one port that supports both channel and network interfaces, relieving the computers from large number of I/O ports. FC provides control and complete error checking over the link.

The next section discusses the methods of managing distributed storage systems.

[16] Fiber Channel Industrial Association – *general reference [19]*

5. SNMP[17]

SNMP (Simple Network Management Protocol) is an industry standard for controlling networking devices such as storage appliances from a single management application. SNMP is a set of network management protocols and functions that communicate using the Internet Protocol (IP) stack. SNMP allows administrators to manage multi vendor network appliances for troubleshooting purposes and also for configurations and health monitoring of the various networks.

SNMP consists of two parts: *Manager* and *Agents*
A manager is software application that runs on a UNIX, PC or Macintosh computer (designated as the management station). It acts as a console or management station through which network administrators can perform functions.
Agents reside on the network on appliances such as storage devices and generate information such as Ethernet addresses, TCP/IP addresses and traffic statistics about the device on which they reside. The information is then stored in Management Information Bases (MIBs). Proxy agents act on behalf of a device that has not implemented SNMP. The primary communication in SNMP is UDP (User Datagram Protocol). SNMP agents listen on UPD port 161 for receiving requests from the manager whereas the manager listens on UDP port 162 for receiving asynchronous traps.

SNMP strictly works in the application layer or user level without modifying underlying kernel states. As an Application Layer protocol in the seven-layer OSI Model, SNMP normally uses UDP (User Datagram Protocol) and defines a method of communication. It creates a client-server relationship. The client application, also called the network manager, makes virtual connections to an application program, called the SNMP agent, running on a remote network device. The database controlled by the SNMP agent is referred to as the MIB (Management Information Base), and is a standard set of statistical and control values. SNMP sometimes extends these standard values with values specific to a particular agent or user requirement through the use of custom MIBs.

5.1 SNMP Architecture

A simplified representation is shown in figure 11. The dynamic port is assigned by the operating system. The SNMP architecture module consists of a collection of network management stations (NMS) and network elements. Network management stations execute management applications, which monitor and control network devices. Network devices are hosts, gateways and terminal servers that have management agents responsible for performing the network management functions required by the management stations.

SNMP is used to communicate information between network management stations and the agents in the network elements.

[17] Cisco SNMP White paper. *Reference [15]*

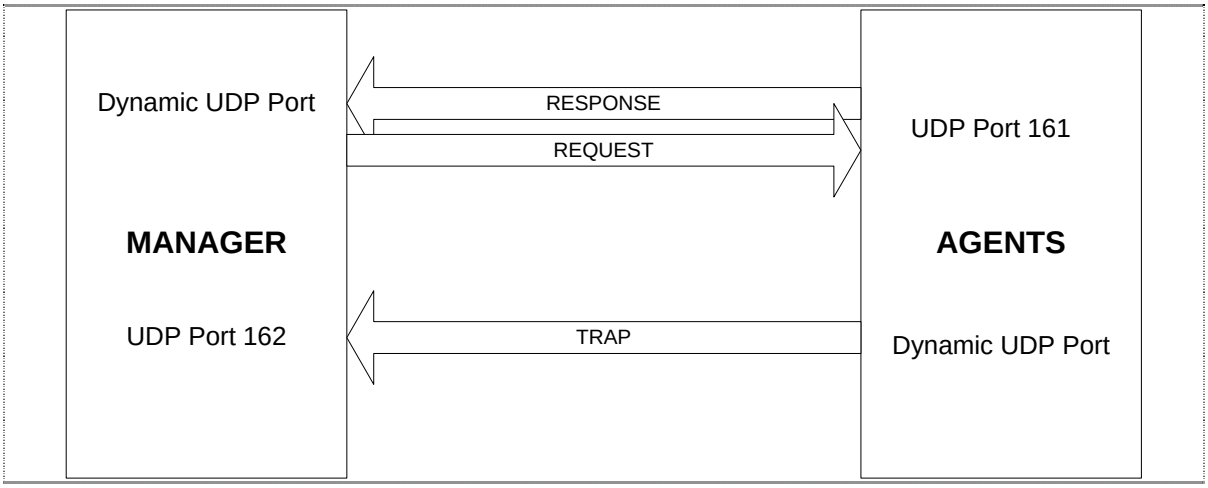

Figure 11: Simplified SNMP Architecture

5.2 SNMP Manager and Agents

According to the SNMP model, a managed device such as a distributed storage system contains software components called agents. This agent then monitors the operation of the storage (managed) device. It does this by maintaining a list of variables called objects, in the *Management Information Base* (MIB).

The MIB represents the operation of the storage device. Agents can be any devices on the network that need to be managed and that have the SNMP protocol and the Management Information Base. Agents monitor the desired objects in their environment; package this information in the appropriate manner, and send it to the management station either immediately or upon request. Information is generated by the Agent, stored in its MIB, and made available to the Manager. Proxy Agents act on behalf of a device that has not implemented SNMP.

A manager program, which normally executes on a network server, exchanges messages with the agent to access the agent's MIB. The manager reads from, and writes to, objects in the MIB according to predefined access privileges that have been assigned to the MIB objects.

SNMP defines the protocols and message formats used to perform the read and write operations; these are called gets and sets, respectively. The Management Information Base (MIB) contains data available to a network management program. Management agents create this MIB so that each machine with an agent will have an associated MIB. The network manager will query this MIB and may even have a management MIB of its own. The management MIB contains general information and the individual MIB contains machine-specific information. The MIB is the definition of the data, or objects, that are stored in the agent for the manager to access.

Agent MIB objects can include values that describe the status, statistics, and general operating parameters of the device. The agent contains the objects that are polled by the network manager and the objects contain data that the manager can collect and display such as:

- Interface Information
- IP datagram
- UDP datagram

Most networking devices that support SNMP support MIB II. As MIB II is a published set of data definitions, any SNMP Manager can access MIB II data. Vendors have also created their own sets of definitions, called custom MIB, so that their own Managers can gather more product-specific information than is available from MIB II.

6. Client-Server Protocols

There are several protocols a NAS or SAN appliances uses for communications with different types of clients.

6.1 Samba[18]

For a distributed storage system that may be running a Unix operating system may have a lot of windows clients. In order for create a client-server relationship between the two; a suite of Unix applications is required to do the job. Operating systems such as Microsoft Windows use *Server Message Block* or SMB for performing client-server networking; hence the Unix Application needs to understand the SMB protocol.
Such an application is called *Samba*, originally developed by Andrew Tridgell in Australia. Samba can run as a couple of daemons in the Unix background and support the SMB protocol.

By supporting this protocol, Samba allows Unix servers to get in on the action, communicating with the same networking protocol as Microsoft Windows products. This enables a SAMBA enriched storage appliance to provide the following features to its windows clients:

- File systems sharing
- Share printers installed on both the server and its clients
- Network Neighborhood browsing
- Authenticate clients logging onto a Windows domain
- WINS name server resolution

In order for this to work, a storage system needs to have two daemons supporting the SMB protocol for SAMBA: *smbd* & *nmbd*. These provide shared resources to the windows clients within the network. They are also sometimes referred to as services.

smbd is a daemon that allows file and printer sharing on an SMB network and provides authentication and authorization for SMB clients.

nmbd is a daemon that looks after the Windows Internet Name Service (WINS), and assists with browsing. It understands service requests by SMB clients such as Windows 9x, 2000 etc.

6.2 Network File System (NFS)

NFS is actually more robust than Samba, but it is also more complicated depending upon what features is used.

From a basic point of view, NFS and Samba are very similar. Both have a client and server application. Both allow a server to share files with clients. Both have clients and servers on almost every platform. The big difference is that Windows PCs have Samba-compatible clients and servers as part of their default network support and Windows

[18] Using SAMBA (David Collier-Brown). *Reference [2]*

requires third party software to support NFS. Conversely, UNIX systems usually come with and use NFS by default with Samba being used to provide file sharing with Windows PCs.

In distributed storage architecture, NFS uses a resource browser capability, which allows an NFS client to lookup a particular Storage appliance running NFS. A program called `showmount`[19] can be used. The `showmount` utility shows status information about the NFS server on the specified host. By default, it prints the names of all hosts that have NFS file systems mounted on that host. We can see what directories have been exported. Samba clients do this as well using NetBIOS but they can also determine what servers are available as well. The `showmount` program must be given the domain name or IP address of the storage server.

NFS works as well as Samba in providing file-sharing support in a closed network that is not connected to the Internet or one that has a good firewall in place. The Linux implementation of NFS tends to be relatively secure by default but it is possible to configure NFS servers, so there are open security holes[20].

The latest Linux kernel has NFS support built-in. The NFS client and server applications tap this support. The NFS server has multitasking support that provides better performance, although this feature is targeted at larger networks.

NFS can be set up by editing the configuration files associated with the NFS support or running applications to mount or dismount an NFS directory. Most of the Linux setup applications, such as `linuxconf` found in Red Hat Linux, can also be used to configure the NFS server and client.

[19] FreeBSD Unleashed (Michael Urban). *Reference [3]*
[20] NFS Security issues – linux magazine. *Reference [9]*

7. Backups and Recovery

As more and more data gets stored, network administrators often need to backup important data on a regular basis. This also often poses as a challenge as the data that needs to be backed up could be that which needs a 24/7 uptime[21]. In the past administrators had no choice but to move all back-up processes during scheduled down periods. For many businesses today, this is also unacceptable and requires data to be available all the time. Distributed storage system has addressed some of these requirements to a certain extent, but of course with a price. Some of the solutions that can be implemented are[22]:

- Backing up a mirrored copy of the data, thereby having one or more instances of that data still available. Following are however some of the disadvantages:
  - This requires systems to already have a mirrored copy of the data. It the system is un-mirrored, a new copy needs to be created for backup purposes.
  - For mirrored systems, during backup of a mirrored copy, the data is vulnerable to losses as there is no more protection on them until the data of the mirror is backed up.
- Implementing locks or disabling some or all parts of the file during a system backup. Though this does not require any additional storage following are a few disadvantages:
  - Use of locks disables some operations on the data during the backup. Certain processes such as changing *metadata* (rename etc.) cannot be done.
  - It does not facilitate consistency between contents of the file, whether on the same or different file systems.
- Creating stable read-only copies of the data. This is sometimes called as a *snapshot.* When a *snapshot* is created, all subsequent writes do get written to the *active* file system. Modifications to the blocks of the accessed file are kept in the holding space after a *snapshot* is created. Access to the *active* file system will provide up to date versions of the file, and access to the *snapshot* copy provides a stable version of the copy. This does not affect file access problems as seen in the previous two methods, however following are some of the disadvantages worth mentioning:
  - Additional storage required, however it accounts to only 2 – 20% of the space because only modified blocks needs to be saved.
  - Performance hit due to accesses via the snapshots
  - Efficient utilities are required for backing up data from one point or the *snapshot* and restoring it from another or *active* file system

As we see there isn't a perfect solution for backup processes. Each one can be implemented at the expense of either performance or storage space. However the third *snapshot* option is still considered a better choice.

[21] Infrastructure supporting the Infrastructure (EMC whitepaper) – *reference [17]*
[22] Data Protection white paper, *general reference [14]*

8. Summary

A storage system's file system is probably the most important component for the system. This along with the overall demands for storage is increasing for commercial server environments. Distributed Storage systems incorporate the tremendously useful journaling file system, which allows a much faster recovery from a system halt condition prior to writing to disk. Older storage appliances used to incorporate time intensive file system checks thereby making data unavailable until the file system check was completed.

NAS and SAN appliances have proven themselves by being the best in the industry for storing and management of digital content. System administrators have a choice of these two technologies for their data. However customers often do bump into the application areas where there are requirements for both type of technologies. In addition to distinctions pointed out in section 2.2, SAN and NAS are chosen for their unique application differences. SAN is capable of providing high performance for streaming application such as high definition video etc, NAS on the other hand has tremendous performance when it comes to applications such as rendering video, data availability etc. Many end users often have a need for both. For example studio and broadcasting companies have their video engineers working on extremely large image or video files and heavily rely upon NAS appliances to render their files quickly and effectively, whereas hosts who need to view these files need the benefit of SAN to be able to seamlessly view that content. Such end users are not experts in deciphering bits per second or bytes per second throughput for reads or writes provided by a given application, and settles for either of the two technologies primarily for one reason: due to lack of resources its easier to manage one particular infrastructure, either NAS or SAN[23].

Though SAN products are the best for streaming data through the wires, implementing a SAN fabric into a network isn't very simple and requires quite a bit of setup work. Each SAN fabric requires fiber channel connection between its own individual nodes. Every host or workstation that attaches to a SAN either via server or directly will require a SAN host adapter. Scaling hosts increases costs and a host adapter needs to be installed in each one of them. NAS on the other hand essentially is a plug and play appliance and depending on the network file system or *NFS* for media access.

In an existing SAN fabric, a user cannot add NAS appliances and have the NAS utilities manage the entire infrastructure – in an attempt to move away from managing the complexities of SAN management. NAS vendors competing in this space have yet to come up with technology that seamlessly integrates into a SAN fabric and *take charge.*

Lastly with the advent of technology we have seen large-scale developments in the storage space, for example hard drive companies have been consistently increasing platter size and access speeds, other areas in this space have not kept up to that pace. Operating systems used today have a lot of shortcoming and have not managed to scale

[23] Byte & Switch Magazine, *reference [10b]*

as fast, thereby hindering overall system performance. This is primarily due to the backwards compatibility to the legacy file system software components.

But needless to say with the current intelligence built into distributed storage systems, they are the perfect choice for users who consider their data extremely precious and who businesses thrive on handling and distribution of such data.

9. Evaluation & Critical Appraisal

The advantages of distributed storage architecture namely SAN and NAS are numerous, to name a few:

- Increased connectivity
- Increased manageability
- Increased flexibility for change and high performance
- Allows multiple hosts to access same files from various operating systems
- Serving web-pages to hundreds or thousands of computers at once

SANs are well suited for dedicated storage such as dedicated storage for applications such as databases, online transaction processing (OLTP). NAS on the other hand is more suitable for general file serving and suitable for applications such as web hosting, CAD or multimedia applications, etc.

As time goes by we are seeing an increased merging between enterprise storage, SAN and NAS. Organizations are evaluation what technology best fits their needs and uses a productive and practical approach by either utilizing either of the technologies or combine them to achieve the information needs of the organization.

This dissertation was written with the following considerations:

- Objective: To provide a detailed explanation of the architecture of distributed storage systems and its technology
- Coverage: Ensuring that all aspects of this technology was covered, namely:
    a. The industry needs for distributed storage
    b. Current technologies such as: SAN and NAS
    c. Their architecture and technology concepts
    d. Their applications
    e. Their advantages and disadvantages
- Accuracy: All factual information used to create this dissertation is either marked or from a list of sources provided in the bibliography. I have attempted to make this to the best of my knowledge free of grammatical and spelling mistakes. However they could be some spelling inaccuracies that may not be rectified yet.

The following considerations may have change the way this dissertation was approached:

- Much of the matter used is from various sources such as magazines, the WWW, at work resource/ experience and academics. All these sources were good for acquiring much of information for the technology. However actual visits to data centers and talking to end-users on their experiences in using such technology would provide more insights on the following:

    - Customer feedback regarding ease of use of this technology
    - First hand info on applications they are used for
    - Issues and benefits the end user has seen by using the product

- Such visits were made after the dissertation was outlined and much of it written by then. An earlier visit would have changed the structure of this dissertation into more in the following lines:

    - The distributed storage technology
    - The limitations with current technology
    - More emphasis on customer experience
    - Potential technology improvements and goals to overcome some of the blocking limitations
    - And the direction, today's vendors of these products, are taking to meet these goals

This dissertation was aimed more at providing information regarding the subject matter, distributed storage systems, and is targeted to a broader audience to enhance their knowledge on the topic. This is exactly what this material would have accomplished. Attempts have been made to make this dissertation concise without too many distracting graphics and at the same time providing an un-ambiguous flow to the information while covering the topics.

Bibliography

References:

[1] Timothy Parker (1997), *Teach yourself TCP/IP in 14 days, Second Edition*. Prentice Hall/ MacMillan Computer Publishing

[2] David Collier-Brown, Robert Eckstien (2003), *Using SAMBA, Second Edition*. O'Reilly & Associates

[3] Michael Urban, Brian Tieman (2001), *FreeBSD Unleashed*, SAM Publications

[4] Gary Field, Peter Ridge (2000), *The Book of SCSI: I/O for the New Millennium*, No Starch Press

[5] Alan F Benner (2001*), Fibre Channel For SAN*, McGraw-Hill Professional

[6] Matt Welsh, Lar Kaufman, (2002), *Running Linux*, *Fourth Edition*, O'Reilly & Associates

[7] Derek Vadala (2002), *Managing RAID for Linux*, O'Reilly & Associates

[8] Behrouz Forouzan (1998), *Introduction to Data Communications and Networking*, WCB/ McGraw Hill

[9] Linux Magazine, http://www.linux-mag.com/2000-04/guru_01.html

[10] Byte & Switch Magazine,

a. Barry University email crash, http://www.byteandswitch.com/document.asp?doc_id=19506&site=byteandswitch
b. Case studies, http://www.byteandswitch.com/library.asp?node_id=72&show_type=cs&view_type=browse

[11] IEEE P802.3ae 10Gb/s Ethernet Task Force,

a. IEEE Ethernet Taskforce. (2002) *IEEE P802.3ae.* IEEE, New York *or*
b. http://standards.ieee.org/catalog/olis/index.html (members only)

[12] Maxtor Corporation Product Roadmap, www.maxtor.com

[13] Seagate Corporation Roadmap, www.seagate.com

[14] Bluearc http://www.bluearc.com/html/library/downloads/data_protection_wp.pdf

[15] Cisco Systems, http://www.cisco.com/warp/public/cc/pd/nemnsw/index.shtml

[16] Network Appliance, http://www.netapp.com/tech_library/ftp/analyst/ar1000.pdf

[17] EMC Corporation, http://www.emc.com/pdf/continuity/emc_crm_wp.pdf

[18] Serial ATA Working Group, http://www.serialata.org/about/index.shtml

[19] Fibre Channel Industry Association, www.fibrechannel.org

[20] Gartner Dataquest, *Worldwide Storage Services Market*, February 2002. Online at: http://www3.gartner.com/5_about/press_releases/2002_02/pr20020219a.jsp

[21] International Data Corporation, *Worldwide Disk Storage Systems Forecast and Analysis*, 1999 – 2004

[22] Advanced Technology Attachment (ATA) Specification (ATA-ATAPI-6) http://www.t13.org/project/d1410r2a.pdf