[Storage Technologies and Theory]

An AN!Speech to TLUG, May 2002

Introduction:

Good day! I wanted to speak with you all today to help you make a more educated choice next time you are faced with designing or purchasing a workstation or server. The single most valuable component of a computer or network is the data. The physical component directly responsible for this most precious item is the storage sub-system. if you agree with me on this then should we not spend the most energy understanding this component of our equipment?

In January a few of you may have been to NewTLUG where I first introduced this topic with mixed success. I over-estimated how much information I could pass on in the twenty minutes I was allocated; I hope today to help clear up what I began then. Also this evening, though my focus is on hard disk storage I would like to cover briefly tape storage and your choices in that arena as well.

In the Beginning:

Let me go back to the very most basics. I will try to be as brief as possible however understanding the fundamentals of data storage is important to fully understand your choices in data storage. If you have trouble following this very abbreviated next section please ask me questions!

The storage sub-system of a computer or network consists of:

1) The physical glass-substrate magnetic platters that actually store the information
2) The read/write heads
3) The controller board of the hard drive that controls the Stepper motors, receive/issue commands from the storage controller and transfers data between the platters and the storage controller.
4) The physical layer used to move the data from the physical hard drive to the storage controller
5) The Storage controller which negotiates reads and writes between the host machine (CPU/memory) and the physical hard drives
6) The data bus of the host computer
7) The processor and memory of the host computer.

This list is an abbreviation and doesn't describe every scenario, simply the most common environments.

Data Storage Fundamentals, or, The Hard and Soft of it all:

Hard:

Before we get into the different types of Hard drives and configurations lets look at how the data is actually stored on and read from the hard drive platter. First, everyone here knows what data is; that data is stored and processed as binary information. We describe binary traditionally as "1" and "0", "on" and "off", "true" and "false". Inside the computer binary is measured electrically as a change of voltage. High voltage, say 5v, is a binary "1" and low voltage, say 0.5v as a binary "0". The actual range is decided on the chip itself but this gives you the idea.

The first and most basic part of a hard drive is the magnetic media. In a hard drive we find one or more "disks" called "platters" that are double-sided. In a hard drive we can have just one platter or we can have two or more platters stacked on top of each other. These platters have a magnetic coating that can be manipulated and read.

So then, how do we measure "1" or "0" with magnetic bits when there is no electrical charge? This brings us to the next part of the hard drive, the read/write head. Simply, to write a piece of data we charge the disk head a certain way and this changes the magnetic material on the drive platter into one polarity or the other. To read data we apply a small charge to the read head and measure the voltage returning from the head as the different polarities pass underneath. This works because if the magnetic bit passing the head is in one alignment, the resistance in the head is low. Just the opposite is true when the magnetic bit is in the other polarity. When this bit passes the head the resistance increases and reduces the voltage returning from the head.

Please forgive me, that was simplified and glossed over but I hope you can understand the concept. Now we are back in the familiar territory of high and low voltage representing our binary "0" and "1". Let me describe the path data takes when it is read from the hard drive.

At the base of the arm the voltage level is measured and converted to binary. This data is then sent outside the drive to the logic controller of the hard drive. Here the data is stored in cache until it is needed. Once the logic board is ready to sent the data to the controller card the data is put onto the data cable and sent to the controller card of the host computer. Once the host controller card has the data it again stores the data in it's cache until the processor or system memory (RAM) is ready for it. Once the signal is given, the data is put onto the motherboard's bus and transferred to the motherboard's chipset that negotiates the final delivery of the data to be processed.

When we want to record data the reverse is true. When the computer is ready to store data the CPU puts the data onto the motherboard's bus and sends it the hard drive's controller. The controller picks it up and stores it in its cache waiting for an opportunity to send the data to the hard drive. Once ready it takes the data and places it onto the data cable where it travels up to the hard drive's logic controller. This time the data is stored in the hard drive's cache until it is ready to write the data to the disk's platter. Once ready the data is placed onto the cable that goes to the read heads where it is converted into a signal that will align the magnetic material into the proper alignment.

Soft:

Now we have a fundamental understanding of how the data moves. Let's take it up a notch now and look at data storage on a more logical level. The surface of the platter is covered with (in modern hard drives) billions of magnetic bits to work with, how do we make sense of it all? We need to arrange this mass of storage into an organized and addressable structure. No matter what file system we use the fundamentals are the same. First up we have to give each side of each platter a number. Lets use an example hard drive with two platters. Obviously we have four sides that we'll number 0, 1, 2 and 3. Each side of each platter has a head so it makes sense then to give the heads the same numbers.

Now we can address the surfaces, lets draw parallel "tracks" on the platters working from the inside out. Those of you who've seen a record can probably picture this quite well. Now we can address a specific track on a specific platter. Let's do one more thing and cut the platter like a pie and give each wedge a number.

Now we can be pretty specific! I can call one specific head, tell it to move to one specific track and wait for one specific slice. This slice we call a "sector" and the size of this sector is usually 512-bytes long (4,096 binary digits, or "bits").

Before I start talking about the fun stuff, does anyone have a question specifically about how data is physically stored, arranged and transferred?

The Fun Begins:

All right, now the fun begins! In modern hard drive we hear three main topics when measuring the capabilities of a hard drive. They are, "Capacity", "Interface" and "External Transfer Rate". I want to spend a little time here because this is the section that just about everyone here will need to know when buying and choosing a hard drive for any system.

Capacity, or A Lesson in Marketing:

First and foremost is the storage capacity of the hard drive, often measured in "gigabytes" these days. The smallest piece of data is a single binary digit called a "Bit". This is a single value of "1" or "0". This is too small to really be useful though so instead we measure data in "bytes". A byte is eight bits long. Why eight? Eight bits give us 256 possible values. In the early days of computing the "ASCII" group assigned a value to each possible combination, which the computing industry agreed on. This common foundation is why raw text can be read on any computer regardless of the operating system. Before anyone shoots me, I know it's only the base 7 bits that are truly standard but for the purpose of this talk please forgive me with saying all eight are standard.

Now seeing that a "byte" is the smallest value of any real use we use it as our foundation for measuring data storage. Just like how we use multiples of 1000 to simplify talking about everyday numbers we use multiples of 1024 to simplify talking about the size of data.

Why use 1024 instead of the easier to remember 1000? Simply put, computers only understand two values, "1" and "0" hence we say computers use a "base 2" number system where we use "2^x" to count and 2^10 is 1024. In everyday speech we have ten digits to work with; 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 and so we say this is a "Base 10" number system. Don't worry if that was too confusing just remember that the closest "base" number to 1000 for a computer is 1024.

In the early days of computers most consumers were pretty savvy about this topic. Thus the stated amount of space on hard drives was as a multiple of 1024 (using Base 2), which was technically accurate. Then some marketing "genius" got a bright idea and alas us techies were condemned to forever explain a growing deviation between the sizes of a hard drive as advertised versus the size of the drive as reported by the computer.

I'm sure everyone here now that one "kilobyte" equals 1024 bytes (or 2^10). The next step is a megabyte which equals 1,048,576 bytes (or 2^20). Now we most commonly see the next step, the "gigabyte", which is equal to 1,073,741,824 bytes (or 2^30). These are the closest base values in computing to their Latin namesakes of 1,000 for "kilo", "1,000,000" for "mega" and "1,000,000,000" for "giga".

What did this marketing demon do to cause us grief? They realized one day that they could legally call a "gigabyte" one billion bytes (10^9) because technically "giga" meant "1,000,000,000", not "1,073,741,824" (2^30). This let the marketing bloke seem like his (or her) hard drives where bigger then their competition by using the "Base 10" 10^6 or 10^9 instead of the "Base 2" 2^20 or 2^30 as a base for dividing a megabyte or gigabyte of storage respectively.

Needless to say the rest of the pack quickly caught on and now the industry standard is to use the historically correct Latin meanings to "giga" and "mega" rather then the more useful and technically accurate meaning of "giga" and "mega". Unfortunately computers haven't changed the way they measure the hard drives capacity so they still divide megabytes and gigabytes the technically correct way using the "Base 2" number system. This also explains why when you look at the storage of a hard drive from within the computer it always seems smaller than what the label says. This gap will only grow as the storage capacity of future drives grow.

For your entertainment, here's how many bytes are needed to make each storage category. Note that I am using the technical meaning of the Latin words!

Term	*Factor*	*Bytes*
Yottabyte	2^80	1,208,925,819,614,629,174,706,176
Zettabyte	2^70	1,180,591,620,717,411,303,424
Exabyte	2^60	1,152,921,504,606,846,976
Petabyte	2^50	1,125,899,906,842,624
Terabyte	2^40	1,099,511,627,776
Gigabyte	2^30	1,073,741,824
Megabyte	2^20	1,048,576
Kilobyte	2^10	1,024
Byte	2^1	1

8 Binary Digits (Bits) = 1 Byte

Descriptions of those capacities

Bit = On or Off, 1 or 0, True or False
Byte = One roman digit or Base 10 number (A, B, C, 1, 2, 3)
Kilobyte = A very short story, half a page
Megabyte = A small book
Gigabyte = A truck full of paper (Call Greenpeace!)
Terabyte = 50,000 trees made into paper and printed
Petabyte = Half the contents of all US academic research libraries
Exabyte = 5EB = all the words people have ever spoken
Zettabyte = As much info as there are grains of sand on all world's beaches
Yottabyte = If you wrote one yottabyte worth of information onto legal size paper the stack would be roughly 25,000 light years high, ¼ the diameter of our galaxy!

Let me leave this topic with an example of how the different representations of storage capacity is making more and more of a difference. Seagate claims to have "the largest-capacity disc drive in the world" at 181.6 GB so I will use it for my example.

Marketing	**Technical**
Raw = 181,555,202,048 bytes	Raw = 181,555,202,048 bytes
10^9 = 1,000,000,000 bytes	2^30 = 1,073,741,824 bytes
Size = 181.6 Gigabytes	Size = 169.1 Gigabytes

Difference = 12.5 Gigabytes! That's more than 7%!

Choices Now!:

So now we've covered all the fundamentals, and you already know what you want to store right? Is a gigabyte a gigabyte a gigabyte? Are all hard drives created equal? Just how are the different RAID levels, well, different? How does fiber and copper stack up?

Foundation:

Hard drives come in two flavours these days called IDE and SCSI. IDE has one interface type and two types of cables. SCSI drives have many interfaces and cables and can be configured in two ways.

IDE:

First off lets look at IDE, which stands for "Integrated Drive Electronics". This is by far the most common type of hard drive you will see. In the earlier days of hard drives the controlled for a hard drive was a separate add in card that supported only the drive it was designed for. The name came about when this controller became "integrated" with the physical disk itself, original naming don't you think?

If you wanted to be specific you could say that SCSI drives ***are*** IDE drives because they too have their controlling circuitry integrated into the drive. Perhaps so, but in the real world today "IDE" is used to describe a particular family of interfaces only. You may hear an IDE devices referred to by the interface standard they meet, "ATA-x", "PIO Mode x", "EIDE", "Fast ATA", or "UDMA/xx".

"IDE" was first standardized as "ATA" by the ANSI group in 1994 (though the standard was introduced in 1990). As time went on and drives grew by leaps and bounds the original standard became inadequate. Seagate developed "Fast ATA" and Quantum developed "EIDE" (Enhanced IDE) as extensions to the original "ATA" standard. Eventually the benefits of both technologies became integrated into "ATA-2".

The original ATA standard was designed only for hard disks. Some here may remember cabling their early tape drives into the very slow floppy adapter. These same people probably also remember cabling their CD-ROMs into their sound cards. Those who worked for bigger companies or had deeper pockets may have been lucky enough to have a SCSI controller that they could use to host their CD-ROM or tape drive on.

Finally the industry caught on to the idea of using the readily available ATA interface for talking to these increasingly more common devices (Tape Drives, CD-ROM/R/RWs, DVD-ROM/RAMs, ZIPs, LS-120s, etc). Unfortunately the ATA standard wasn't compatible inherently so the "AT Attachment Packet Interface" was developed. This standard allows the commands needed by these devices to be transferred to the device using the standard ATA commands.

This is also why older OS' couldn't see a device using ATAPI until special drivers where loaded. As time has gone on motherboard BIOS' have become more intelligent and can now even boot off these devices like the CD-ROM. Eventually ATAPI was integrated into the ATA-4 standard.

Configuring IDE:

For those who've opened a computer you've noticed that IDE devices have three main jumper setting; Master, Slave and Cable Select. The ATA standard allows for two devices per IDE channel meaning then that we can have two devices plus the controller all connected on the same cable. Obviously we needed some way to tell which device we were sending data to or need data from.

The most common way to do this is to set the first device (or only device) as Master and connect it to the end of the ribbon cable. The second device would then be set via it's jumper as Slave and cabled to the middle connector on the ribbon cable. Is the position of the cable connectors relative to the setting of the IDE device important? Technically yes but with older 40-pin/40-wire ribbon cables you could usually get away with any configuration

If you've built or worked with a relatively modern computer lately you inevitably noticed the snazzy new cable that came with either your UDMA/33+ hard drive or motherboard. Why the new cable? What was wrong with the old one? The original 40-

pin/40-wire ribbon cable was fine for the original slower speeds of older hard drives but the ribbon cable is not terminated and this causes signal bounce and cross-talk that can corrupt data. I'll talk about signal bounce, cross-talk and termination in better detail shortly when we get into SCSI.

As modern hard drives reach higher and higher speeds the signal bounce and cross-talk found in the older 40-pin/40-wire cable became too great to ignore. Something had too be done to lessen the distortion while still keeping costs down. From this need the 40-pin/80-wire ATA cable was born! This cable is physically compatible with the older style 40-pin IDC connectors you've all seen but is much better at carrying the higher speeds.

So then, how does a cable with 80 wires but only 40 pins help? Just what are those 40 extra wires connected to? Well for starters they are indeed connected to something; they are grounded. How does this help then?

Time to again get fundamental for a moment! As I mentioned earlier data travels through a computer in the form of electrical charges. In physics when an electrical charge passes over copper you get a magnetic field. Likewise when a magnetic field passes over a copper wire an electric charge is created. This is how electric motors and generators work. Remember as well that a computer distinguishes a "1" from a "0" based on the voltage level of the signal.

How does all this figure into the new cable? When we send a voltage down a data wire it creates a small magnetic field. This magnetic field can pass over the wire next to it and increase the voltage on the neighboring wire. If this effect is strong enough it can drive the signal on the next wire high enough that the computer or logic board sees a "1" when it should have seen a "0". Voila, your data has been corrupted.

The new cable specified in the ATA-4 standard tries to deal with this by placing grounded wires between each data wire on it's 80-conductor ATA cable. This way the stray magnetic fields are "absorbed" by the grounded wire before it has a chance to interfere with the next data wire. ATA-4 didn't require this cable be used but it was strongly recommended. ATA-5 and up however does required this cable in order to function at full speed.

It is also now very important to be sure to put the master device at the end of the data cable and the slave, if present, on the middle connector. The ATA-4 standard even calls for colour coding to help ensure you remember this. You'll notice now that all these new cables have blue, gray and black connector. The blue connector attaches to the mainboard or add-in IDE controller (for example an IDE RAID card), the gray cable connects to the slave device and finally the black connector attaches to the master device.

There is another feature to these cables that you'll find interesting. All these cables support the "Cable Select" setting available on IDE devices. This was supposed to be implemented in the traditional cables but for many reasons was mostly ignored. Pin #28 on an ATA IDE cable is grounded on the mainboard and on the gray (slave) connector but is open (not grounded) on the black (master) connector. If a device is jumpered to

"Cable Select" then when the device powers up it looks to see if Pin #28 is grounded. If it is, it configures itself as a slave device. Likewise if the pin is not grounded then the drive configures itself as a master device. How simple!

There is one last thing that the group who developed had to take into account when developing this new cable. What happens if someone used an old-style 40-pin/40-wire connector? If you tried to transfer data at the speeds proposed then data corruption would likely crash the computer. That wasn't acceptable so some way of telling if the right cable was installed was needed.

They did this by pulling pin #34 to ground on the new cable. On older cables pin#34 wasn't grounded so all the computer needs to do is check if pin #34 is grounded or not to tell if the right cable is being used. If it isn't then the transfer rate is reduced to a safe level. You'll even notice many better mainboards will announce during POST that the wrong cable is being used!

One more note on IDE cables, both the traditional and 80-conductor ATA IDE cables have a maximum cable length of 45 cm! There are many manufacturers who make longer "IDE" cables but these cables are defiantly not recommended. In the best scenario you will suffer a performance hit, though quite likely you will see data corruptions or unexplained system crashes.

Marketing IDE:

The last thing I want to cover regarding IDE is how the speed and performance of a hard drive is advertised. The advertisements often proudly state the speed on the bus that their drive is compatible with. This is usually done in a way that makes most people think that's the speed they'll move data at when in fact it's usually not even close.

Let me beat up on Maxtor because I find they use this confusion to their benefit the most. They proudly state their new "Fast Drives ATA/133" (which isn't even a ratified standard yet) with the logo on the right.

First up, it's not the drive that's actually fast; it's the interface that is. What you need to look for is either the sustained average external transfer rate or the sustained external transfer rate at the inside and outside edges of the hard drive's platters.

Let me again compare using real world examples:

The Maxtor "DiamondMax/Plus D740X" line, one of Maxtor's UDMA/133 "Fast Drives", presents these specs:

Fig x.x, Specifications of the Maxtor DiamondMax/Plus D740X

Data Transfer Speed (MByte/sec, max)	
To/From Interface	133
To/From Media	54.2
Sustained at OD	44.4
Sustained at ID	24.2

http://www.maxtor.com/products/DiamondMax/DiamondMaxPlus/DataSheet/D740X_datasheet.pdf

Compare those specs to a "standard" UDMA/100 Maxtor DiamondMax/Plus 60 specs:

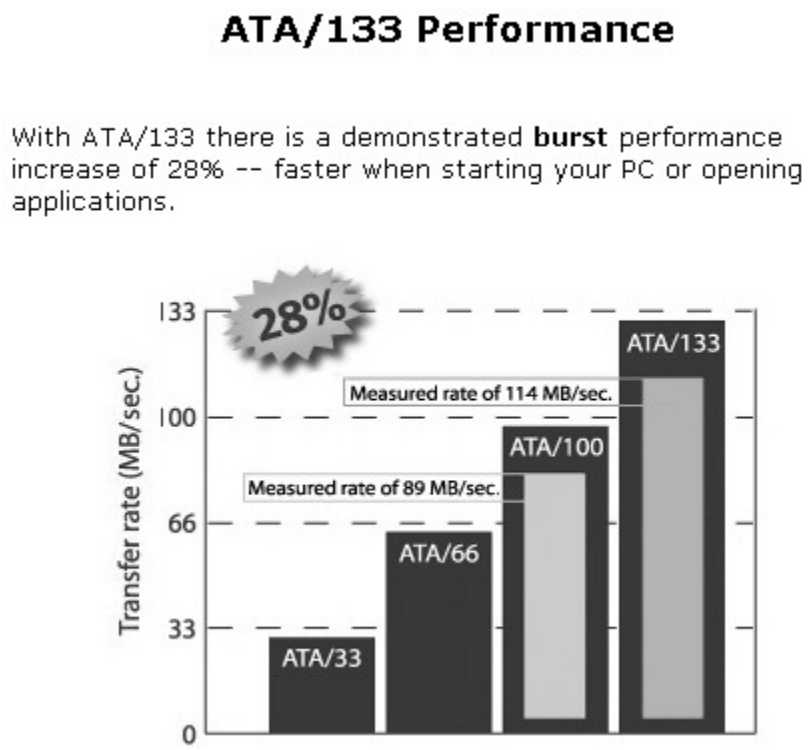
Fig x.x, Specifications of the Maxtor DiamondMax/Plus 60

Data Transfer Rate	
To/from Interface	up to 100 MBytes/sec
To/from Media	up to 57 MBytes/sec

http://www.maxtor.com/products/diamondmax/diamondmaxplus/FullSpecs/dmp60spec.pdf

Notice two things; first off the DiamondMax/Plus 60 doesn't list the drives transfer rates on the inner and outer tracks of the platter. Secondly though, and more telling is the average transfer rate from the "Media" (drive platters). Assuming Maxtor didn't change the way they test their own drives it would seem the "slower" DiamondMax/Plus 60 is actually faster than the "Fast Drive" DiamondMax/Plus D740X.

Fig x.x, Maxtor's stated Burst Rates



Let's look at something else: Burst Rates. This is another great opportunity for marketing people to inflate the performance of their drives. Modern hard drives have dedicated memory (RAM) on the logic board that the drive uses to store cached information. The controller on the drive tries to guess what information will be requested shortly and fetches it from the platter and stores it in cache. If it guessed right then you have a "cache-hit" and the data can be pulled from the drive's cache and passed to the interface MUCH faster then it would have otherwise taken to get the data off the drive's platters.

http://www.maxtor.com/Maxtorhome.htm

Let me state up front, Cache is a good thing and the more cache the better because the more data you can store in cache the more likely you are to have a cache-hit. That said, cache it not a good way to measure overall performance. Much more often than not the cache doesn't contain the data needed and the drive must fetch data from the platters anyway negating any benefit the cache may provide. This stated, Maxtor decides again to emphasis the performance of their UDMA/133 drives by quoting maximum transfer rates based on *burst* rates.

A final note on maximum transfer rates on the IDE bus. Why worry so much if the sustained external transfer rate of the average hard drive is so much slower than the speed of the system bus? The first answer is to allow for the performance increase in future drives. Secondly, we have two devices per IDE channel that have to share the bandwidth on the bus. If you add the maximum sustained transfer rate of both devices on an IDE channel and the resulting number is less than the maximum throughput of the bus then you will never have a performance bottleneck on the IDE bus even if both devices are going all out.

Finally there is one more number to pay attention too: platter rotation speed. Remember earlier I explained how the heads are moved over a specified track then wait until the right sector passes underneath? The time the drive spends waiting for the right track to pass is wasted time. A very good way then to improve performance would be to increase the speed that the platter rotates so that it doesn't have to wait as long. Well this is exactly what the industry has done. Today there are drives on the market that spin at an astounding 15,000 rotation per minute (rpm)!

Be VERY careful when installing a drive that has a high rpm rating. Most IDE drives rotate at either 5,400 or 7,200 rpm. Centripetal force will try to keep the platters in the same place even when you move the casing of a hard drive. If the read/write head moves and the platter tries to stay in place there is a decent chance to two will make contact and you will have a true crash, immediately destroying the hard drive and it's data. Install the drive either horizontally (preferred) or vertically (only if you need to), never an angle in between. Be sure also that the drive is firmly mounted and NEVER move the drive while the platters are spinning. Be sure also to give the drive plenty of time to spin down before moving it. Keep in mind this means not moving the computer at all for about 30 to 45 seconds after the power is turned off.

Look at table **x.x** at the end of this document for a side-by-side comparison of modern hard drives. These drives represent each company's best IDE drives as of the time I wrote this.

SCSI:

Let's look at "SCSI" now! First up, the performance and marketing considerations I just spoke about regarding IDE also apply to SCSI!

Now then, why do we bother having two interfaces? What is it about SCSI that's worth the trouble? As a matter of fact, SCSI has a lot of advantages. Trick is for most users the advantages to SCSI are simply overkill. That said you don't need to venture very far at all into the server world before you realize that IDE simply isn't up to the task. This is the arena where SCSI shines!

"SCSI" is short for "Small Computer Systems Interface" and is most often pronounces "skuzzy" but it's technically more correct to say "S-C-S-I". That though, is a debate with no end; go with whatever is easier for you to say. Consider the alternative "SASI" (Shugart Associates Systems Interface) which was the original name given to it by the original developer Alan Shugart back in 1979. He later went on to start the hard drive company "Seagate Technologies".

SCSI was original defined as "SCSI-1" in 1986 even though the group to standardize SASI first began its work in 1982. It was designed to be much more capable and expandable than IDE. IDE was originally an interface to and an extension of the 16-bit ISA bus. Later it moved on to the PCI bus but remained mostly just an interface non-the-less. SCSI on the other hand was designed from the ground up as a system bus in and of itself. For this reason you'll notice SCSI adapters are often listed as "SCSI-to-PCI adapters".

SCSI was also designed from the ground up to support both hard drives and peripherals so no special software was needed to support these devices. This allowed it a flexibility IDE could only dream of! SCSI also supported up to 7 additional devices off of a single SCSI adapter and connected easily to both internal and external devices simultaneously.

Today SCSI has reached its third and most prolific version called "SCSI-3". This version supports speeds up to 320MB/sec over copper and 15 devices! Please note that SCSI-3 allows for Fiber Optics as the physical transport that ups the number of devices per channel to 120 and increases maximum distances to 10km with speeds of up to 400MB/sec using dual-ported devices. I won't get into fiber today however because it could quickly become a topic in and of itself.

If enough people are interested I would be happy to present again at some future time with a focus on Fiber channel, SANs and NAS. Please speak to me after this presentation if you are interested.

The price of SCSI devices of course reflected the added benefits of the SCSI protocols. This has kept SCSI out of most computers because for most users IDE was adequate. For high-end workstations and servers however IDE falls flat on it's face.

Keeping in mind the most common users of SCSI the drives themselves are built on the newest technology and that inevitably translates to yet a higher cost to produce. SCSI drives also require a special adapter card that itself often costs far more than the best IDE drives on the market. Finally the price is driven up by the complexity of installation. There are several factors to consider and a little study is definitely needed to get it right. Where IDE has one common interface type and only two cables to choose from SCSI drives can have multiple different types of connectors and cables some of which are physically the same but electrically incompatible!

Standards:

SCSI-1 is now obsolete and you are unlikely to us it unless you are like my friend Lance who seems to relish building ancient machines! SCSI-1 came in two flavors; the far more common Single Ended (SE) and the far rarer High Voltage Differential (HVD). SCSI-1, being the first ever flavor of SCSI, offered no special options (it was special all by itself). Using SE devices you could have a cable up to six meters long. If you could afford the extra cost of HVD you could have a cable as long as 25 meters long!

SCSI-1 called for two cables; a 50-pin ribbon cable internally and a D-Sub 50 pin externally. Apple Computers however decided to strip off the ground wires that were called for to allow them to use the more readily available and cheaper D-Sub 25 connector. Though this was never an officially accepted SCSI connector it was used up until just a few years ago (remember the SCSI Zip drive?). This was a problem because the D-Sub 25 is the same connector used by the parallel port though they were electrically very different!

The internal cable was similar to the original 40-pin/40-conductor IDE cable except it was 50-pin/50-conductor. This cable was also used though SCSI-2 until just recently. The largest hard drive I've found that uses this cable was a 9GB drive from Seagate and IBM though I am sure that all major suppliers had drives this big that still used the IDC50 cable.

SCSI-2

There is another benefit to SCSI proper before we go into RAID and that is spindle speed. Remember how IDE drives today spin up to 7,200rpm? SCSI drives today only start at 7,200 rpm and get as fast as 15,000rpm! Remember I mentioned it's more expensive to build a SCSI drive, accuracy and reliability at those speeds doesn't come cheap!

RAID:

So now you see the basic differences between IDE and SCSI. Well, what happens if all the goodness of SCSI just isn't enough? "RAID" stands for "Redundant Array of Inexpensive Disks". It's funny to hear that when you consider a decent entry-level server with RAID starts around $6,000! You need to remember that the term was coined back when disk space was still measured in $$/MB.

You don't technically need SCSI drives to implement RAID; in fact there is a trend in the performance home market right now for IDE drives to be used in RAID arrays. I wouldn't recommend it yet for people needing high-levels of reliability or processing power though.

So what is a RAID array? Being here you probably have enough interest in computers to have heard of RAID but unless you are slightly obsessed with hard drive technology you probably haven't learned much about it. RAID has been, and to a large part still is the domain of higher-level servers.

RAID describes three main abilities that can be implemented either alone or in combination to best fit various scenarios. These features include "stripping", "mirroring" and "parity".

Stripping, known as RAID level 0 or RAID0 is the process of using two or more drives for simultaneous writing and reading. When a file is to be written to a stripped array the data is divided into chunks and written to the drives in the array at the same time. As a loose example you can take a 10MB file and write it to a RAID0 array with two drives in roughly the time it would normally take to write a 5MB file (twice the speed). The same 10MB file could be written to an array with five drives in roughly the time it would have taken a 2MB file to be written to a single drive (five times as fast). Calculating the actual speed benefits isn't so cut and dry because of other overhead but you get a good idea.

Next up is "Mirroring" or RAID1. As its name implies, two drives are mirror images of one another. If one drive fails the data is safe thanks to the second identical drive. The down side is that 50% of the physical hard drive space is wasted.

Finally we get to "Parity", used in RAID3, 4, 5 and 6 but most popularly in RAID5. Remember in math class you asked "where will I ever use this in the real world?" Well my friends, Boolean algebra has allowed us a very efficient was to protect data. Lets use a RAID5 array for this example but first let me describe a RAID5 array.

In a RAID5 array you need a minimum of 3 disks. The more you add though the better performance you gain and the more efficiently you use your disk space. The trade off is you need an increasingly more powerful RAID controller and that translates to a higher cost. In a RAID5 array performance is increased by stripping data across the available drives (RAID0). In a RAID0 array though a single disk failure will destroy all the data because part of just about every file is on each disk. Parity is added in RAID5 to deal with this.

Parity works by taking the data on each disk and using the Boolean "XOR" cumulatively to come up with parity data. This last piece of redundant data can be used to rebuild any one piece of missing data in an array (one failed drive). For this reason RAID5 is described as making use of N-1 disks in the array. In the minimum three drive array we have a 33% waste of space (3-1 = 2 effective disks). In a five-disk array we increase our efficiency to 20% wasted space (5-1 = 4 effective disks).

The down side to RAID5 is that when a failure occurs read times would drop, potentially by a marked amount. This is because for every byte of data being requested in a read the RAID controller must first run the XOR on the remaining good data plus the parity byte before knowing what the missing data is.

I am sure by now that my 20 minutes is almost up, thank you for listening to me today and I hope that if this wasn't too fast or confusing and it at least wet your appetite to learn more about hard disk technology. I will be doing a presentation at March's TLUG meeting where I will go into much more detail about underlying disk technology and how it relates to servers and Linux. If you would like to learn more I encourage you to join me then.

Does anyone have any questions?

Thank you all very much for your attention! The paper for this talk will be available on my web site as soon as some work is done. At that time there will be links that will allow you to study this and other topics in greater detail. Good night!

Notes:

Table Comparing IDE (120 GXP) vs. SCSI (73LZX) vs. SCSI (36Z15)

	IBM 120GXP (IDE)	**IBM 73LZX (SCSI)**	**IBM 36Z15 (SCSI)**
Capacity/Spindle RPM	120GB / 7,200rpm	73GB / 10,020rpm	36GB / 15,000rpm
MSRP in CND / $/GB	$572.52 / $4.77	$1,518.79 / $20.81	$1,097.56 / $30.49
Devices per Channel	2	15	15
Bus maximum bandwidth	100MB/Sec	160MB/Sec	160-320MB/Sec
Data Buffer	2,048KB	4,096KB	4,096KB
Sustained Data Rate	23MB/Sec to 48MB/Sec	29MB/Sec to 57MB/Sec	36MB/Sec to 53MB/Sec
Average Seek Time	8.5ms	3.0ms	2.0ms
Error Rate	1 in 10^13bits (1 in 1.2TB)	1 in 10^14bits (1 in 11.3TB)	1 in 10^16bits (1 in 1,136.9TB)
Rated Start/Stops	40,000	50,000	50,000
Power-on hours/month	333h (11h/day)	732h (24h/day)	732h (24h/day)

RAID Levels:

Level 0 = Stripping, not a "true" RAID level because there is no redundancy. RAID0 is designed to increase performance by cutting data into uniform sized blocks and sequentially writing them to the disks in the array. The total capacity (T) is measured by

multiplying the numbers of disks in the array (x) by the storage capacity of the smallest drive(s). T=x*s

Level 1 = Mirroring, the oldest type of RAID. Very popular to it's ease of use and high-reliability. Write times can be slightly diminished because of the need to write the data across twice the number of disks but a good hardware RAID controller often negates this. Read times are quicker; almost double the speed because the data needs to only be read from one drive. The performance hit during a failure is minimal and the processing power needed to implement RAID1 is small. The main drawback is the initial cost to configure because of the inefficient use of disk space. RAID1 requires an even number of drives. The total storage capacity (T) is measured by multiplying the number of drives in the array (x) by the capacity of the smallest drive (s) and dividing by 2. T=(x*s)/2

Level 2 = Defined but rarely used because it requires special hard drives that manufacturers are reluctant to build. RAID2 splits data at the bit level and stripes it across the data disks. For each strip of data Hamming ECC data is generated and stored on multiple parity disks. The main advantage of RAID2 was its ability to correct single bit errors "on the fly" because both the good data and the parity data were read an analyzed on every read. This Error Correction Code (ECC) is now a standard feature within hard drives, negating most of the benefits of RAID2 right there. RAID2 also suffered from modest performance and reliability when compared to other RAID levels, particularly when the number of disks needed is considered. A standard RAID2 array called for 10 data disks plus 4 parity disks or 32 data disks plus 7 parity disks.

Level 3 = RAID3 is still used today but isn't as popular as other levels. RAID3 implements striping with parity and uses a dedicated parity disk. The data is split into blocks usually smaller than 1024 bytes allowing it to make efficient use of the stripping ability. The downside is RAID3 has a single parity disk, which causes a bottleneck when there are many I/O transactions at the same time. RAID3 is best suited for environments where redundancy is required and most of the disk access involves large file read/writes like you would see creating multimedia. Total capacity (T) is measured by multiplying the number of drives in the array (x) by the storage capacity of the smallest disk (s) minus one. T=(x*s)-1

Level 4 = When they wrote that song "Stuck in the Middle with You" they must have been thinking of RAID4. Like RAID3 and RAID5 it also stripes data across disks with parity. Like RAID3 it uses a dedicated parity disk that can be a bottleneck but it uses larger data blocks like RAID5. The larger data blocks improve the performance when there are a high number of simultaneous I/O transactions but makes less efficient use of stripping. You can calculate the total storage like you did in RAID3, T=(x*s)-1

Level 5 = Perhaps the most popular RAID level used when both performance and reliability is required. RAID5 stripes both data and parity information across all the disks in the array removing the bottleneck seen in RAID3 and RAID4. It uses larger data blocks than RAID3 making it best suited for multiple simultaneous I/O transaction (web server anyone?). During a failure the performance will suffer more than if just the parity disk failed in a RAID3 or RAID4 array because read/writes will require an analysis of parity data to rebuild missing data. Total storage capacity is calculated like RAID3 and RAID4, T=(x*s)-1

Level 6 = RAID6 is very similar to RAID5 except that two blocks of parity information is recorded allowing the array to survive two simultaneous drive failures. This isn't a popular RAID level though because the ability to have hot spares in a RAID5 array means the time when the array is vulnerable is very short. With a hot spare in a RAID5 array you don't suffer the write performance hit seen it RAID6 where two separate parity blocks must be calculated and recorded on each write. You can calculate the total capacity of a RAID6 array similar to RAID5 except you lose two disks to parity information, T=(x*s)-2.

Level 7 = This isn't a true RAID level in that this RAID level is proprietary meaning the controller and standard is owned and controlled by one company. They try to solve some of the problems of RAID3 and RAID4 but details are sketchy. Not a recommended RAID level.

Combining Levels = Not long after RAID began life did people start wondering about combining RAID levels to get the benefits of two levels. Before I talk about those levels I want to make a few comments. The order in which the RAID levels are noted *is* important. For example RAID 10 *is not* RAID 01. In this example, RAID 10 is an array of stripped mirrors as apposed to RAID 01, which is a mirror of stripped arrays. Be careful when faced with RAID 53, this can be RAID 03 or even 30, depending on it's implementation.

Level 01 and 10 = In RAID 01 two striped arrays of equal size are created then data is duplicated (mirrored) on each array. This provides good fault tolerance by being able to survive multiple disk failures so long as they are on the same array. Performance is also good because of the speed benefits of writing to a striped array and the speed benefits of reading from a mirrored array.

RAID 10 creates mirrored pairs then stripes the data across the pairs. This provides a slightly higher level of reliability because the loss of a drive only affects the mirror it belongs to. This helps minimize degradation and rebuild time because less data is mirrored per set.

Both RAID 01 and RAID 10 have the benefit of redundancy without the overhead of parity. Both also require a minimum of four disks and the total number must be even. Calculate the total capacity of the array like you do in RAID1, T=(x*s)/2.

Level 03 and 30 = Hope you have done your mental Yoga; these combinations are probably the most difficult to mentally picture.

RAID 03 is built by using stripped sub-arrays in a RAID3 array. In this configuration the byte data and parity is stripped across multiple RAID0 arrays (minimum 3 RAID0 arrays each with a minimum of 2 drives). This gives performance levels closer to RAID0 but a little slower because parity still needs to be calculated and written. Assuming all the disks are the same size the total capacity (T) is measured by calculating the capacity of each RAID0 array (ra) and then multiplying the number of RAID0 arrays (rb) minus 1, T=ra*(rb-1).

RAID 30 is just the opposite, stripping data across two or more RAID3 sub-arrays. This level is the more popular implementation of combining block striping with byte stripping and parity. It provides better fault tolerance and performance because each RAID3 sub-array in the RAID0 stripe is independently protected. It also makes more sense to byte-stripe blocks of data because you are making smaller pieces out of larger chunks of data. Assuming again that all the drives in the array are the same size you can calculate the total capacity of the array (T) by multiplying the number of drives in the RAID3 sub-array (ra) minus 1 then multiplying the number of sub-arrays in the stripe (rb), T=(ra-1)*rb.

In both cases RAID 03 and RAID 30 provide the best transfer rates but still suffer from bottlenecks when the number of simultaneous I/O transactions increase. These configurations are best suited for applications where large files are used by a few people (like in multimedia) and regular RAID3 isn't fast enough.

Level 05 and 50 = Apply the differences discussed between RAID3 and RAID5 over what was just said about RAID 03/30 and you have RAID 05/05. The main difference between RAID 03/30 and RAID 05/50 is that RAID 05/50 increases performance under simultaneous I/O transactions by using larger blocks of data and by removing the bottleneck of a single parity disk. This makes RAID 05/50 better suited for use in scenarios where many users are requesting simultaneous read/writes to the array and RAID5 alone doesn't provide enough speed. Calculate total capacity as you did in RAID 03/30.

Level 15 and 51 = It doesn't get any more reliable or inefficient than this! RAID 15/51 is the most reliable method of storing data in a single server, combining the raw availability of RAID1 mirrors with the performance and reliability benefits of RAID5. Realistically if this level of reliability was required you would be interested in high availability servers where everything, not just the disk array was redundant.

References:

· The "T10" committee responsible for developing the SCSI standards:
`http://www.t10.org/`

· The "T11" committee responsible for developing the FC, HIPPI and IPI standards:
`http://www.t11.org/`

· The "T13" committee responsible for developing the ATA standards:
`http://www.t13.org/`

· "The PC Guide", by Charles M. Kozierok. THE site on the web for double-checking yourself on computer hardware. If he ever reads this, "Thanks!":
`http://www.pcguide.com/`