

Help!

My NT Server

Is Too

Slow

Disclaimer: This paper was written in 1998 and is therefore not up to date with the newest hard drive technology. However, it does correctly predict what is happening in the storage arena today. Today's hard drives put a lot more data on each track and therefore are able to provide better I/O than the drives that were used in testing for this paper. They are a lot less expensive too. IDE have continued to close the gap on SCSI but SCSI is still the hard drive of choice for most servers. The performance monitoring sections and the sorting sections are still as relevant today as they were in 1998. Please use the information to any advantage you can and be aware that SAS does not provide any hardware support from its Technical Support Department.

Table of Contents

Introduction	3
Section I	
Help, My NT Server's too Slow	4
What is the Goal of Performance Tuning?	5
Understanding how hardware speeds relate to each other.....	5
Hurray! We've found our bottleneck and achieved the Goal of Performance Tuning, haven't we?	6
Finding the Bottleneck	7
The Objects and Counters	11
Memory Objects and Counters	11
I/O Objects and Counters	13
CPU Objects and Counters	14
Section II	
I/O Subsystems	15
The Hard Drive	17
RAID	24
RAID Definitions	
What Levels are there and what should you use?.....	25
RAID Array Implementation	27
RAID Discussion	28
Section III	
The Test	30
Sort Times	31
Testing Summary and Comments	32
The Biggest Difference	34
The Big Discovery	
Affects of Memory on Sorting	35
Conclusion	38
Appendices	
Appendix A	
Performance Tips	40
Appendix B	
Tuning Memory on a Windows NT 4.0 Server.....	45
Appendix C	
Feedback on a hardware suggestion.....	48
Appendix D	
Alternate Resouces.....	40

Introduction

“What seems to be the matter Sir?”

“Well, when I start it up it goes cough, pow, bang, bam, sputter sputter, ka-bam. And then after I finally get it started it kind of lurches down the road. Gets to be real annoying especially trying to hold my cup of coffee in the mornings. Gets all over me not to mention that it’s hot. I’ve received 9th degree burns on my hands and had to be treated at the hospital emergency room yesterday. Is there anything you can do?”

“Sure, we can have a look. Most likely all you need is a good tune-up. When did you take delivery Sir?”

“Uh, well, I really don’t recall. Um, let’s see. It was just this week sometime. This is a brand new car. I expected it to run for a long time before it needed a tune-up.”

“Well, Sir. Sometimes they don’t come from the factory tuned the way they need to be for our part of the country. But that’s OK. We have an excellent staff of factory trained mechanics here and we’ll have your new car tip top in no time at all. Our shuttle bus is getting ready to leave if you need a ride to work. We’ll call you this afternoon and send the shuttle to pick you up.”

Has the above scenario ever happened to you? How about if we were to insert server in the place of car? Oh, yeah and take the burn part out? And the ride down the road part although you may have the feeling you got a ride down the wrong road from the salesman who sold you that thing.

Truth is, most servers do not come already tuned to do the work you need them to do right out of the box. Also, how many companies have trained NT Administrators who can tune servers so that they can provide the best performance for the needs of that company? For many small shops, this is a big problem. Servers arrive, are set up quickly, go on-line and are never properly tuned for the job they’re expected to do. Another problem is in sizing. If you leave it up to the salesperson, they’re going to get it wrong most of the time unless they have done a thorough job of analyzing the workload of their prospective client’s site. And even if the server is sized correctly, the needs of most companies keep changing. This changes the server’s workload as time goes by.

In the following paper, the Bottleneck theory is used as a way of isolating the factors that are hampering performance in the small server arena. You will also find that I/O is addressed in depth as it is seen as one of the biggest bottlenecks to SAS System users. Testing was also done in an effort to try to find ways to increase the performance of small NT Server systems. During the testing some surprises were encountered and are also presented. Finally, appendices were added to offer some quick tips and to add more information on some of the factors governing performance.

Section I

Help, My NT Server's too slow

They say there's good and bad in everything. What comes to mind when you hear the words "Hardware Marketing"? What comes to mind for me here in Technical Support for the SAS System on Windows NT servers, is a nightmarish vision of the ads plastered all over the PC magazines proclaiming that this or that company has just the right machine for your NT Server needs. Prices starting as low as \$2,400.00 for machines that are able to do everything including serving you breakfast in bed abound in this world of marketing hype. For us, there is no lack of understanding that to an IS department head this looks oh so very enticing. Thoughts of moving away from that mainframe, which cost an arm and leg to operate, jump right off those advertisement pages and ignite those 'got to have it' fires. But is it a wise decision to fall for all this marketing hype? And what do you do when that wonderful 'low cost NT server solution' fails to meet your expectations?

"Who you gonna call?"

Understandably, I hear some complaints about performance on Windows NT servers. Most of the problems are found on low cost servers and workstations. But it's not enough to tell you this. As a partner in your business solutions, SAS Institute, and I feel an obligation to give you as much information as we can in order to help you tune your server or workstation to achieve a higher level of performance.

I do not have unlimited resources with which to work. Nor can I test every hardware combination or promise that I can help you achieve better performance. Despite this, I have put together a lot of useful information that hopefully you can use to increase your hardware knowledge and help you achieve better performance on your systems. First, I will take a look at the most important hardware components and how they work. I will try to help you understand what the slowest components are and why they are slow. Then I will take a look at how you can analyze your system and find out what's holding it back. I will also take a look at some of the testing I have done. Finally, I will provide a list of other resources that can be utilized in performance tuning. And I will explain how to use the Sortsize option to increase your sorting speed. Equipped with all these new tools and knowledge, the goal of better performance will hopefully be within your reach.

Some of the suggestions in this paper may require additional upgrades to your equipment. I have tried to maintain the perspective that you have already invested lots of time and money and that it would be OK with you if you didn't have to spend any more. My dad used to say though "Nothing ventured, Nothing gained." So, lets get started and see if we can put it all together.

What is the Goal of Performance Tuning?

To IS management it might be defined as 'Another means of creating more unnecessary jobs in our department' (computers should come from the factory already tuned anyway, right?). Smaller companies may see it as another way the independent consulting companies 'prey on them'. And taken altogether it's a plot to increase the number of computer related jobs for their buddies by the operating system developers, right? Of course, none of these are true. Hopefully, everyone knows that the goal of performance tuning is to 'makes the things work faster'. But how is this achieved?

The simplest answer is through eliminating the bottleneck. And what is a bottleneck? A bottleneck is defined by Webster's Ninth New Collegiate Dictionary as:

- '1. a: A narrow route b: A point of traffic congestion
- 2. a: A condition or situation that retards or halts free movement and progress
b: IMPASSE'.

On a computer, it could be said that a bottleneck is a condition in which a part of the system works so slowly that it holds up the rest of the system. There are three potential bandits that can serve as the bottleneck in a server or workstation. These potential *bottleneck bandits* are part of the main hardware and are the most important parts of a computer. Listed in order of importance they are:

1. CPU/s or Processor/s

The CPU is the main brain of the computer. It does all the processing.

2. Memory (Also known as Physical Ram)

Memory is a storage area that data are moved into while waiting to be processed by the CPU.

3. I/O Subsystem

This consists of the hard drive/s or RAID array/s in a computer system and the controller/s which are used to connect them to the computers motherboard.

I'll stop referring to the above hardware as *bottleneck bandits* and instead call them components from now on. After all, what we're going to try to do is understand these components and hopefully learn how to keep them from becoming *bottleneck bandits*.

Understanding how the hardware speeds relates to each other

In order to understand how the above components affect the performance of a computer, you must understand how their speed is related to each other. The three measurements of speed most often used are:

1. MHz or Megahertz
(Used to measure the speed of the CPU)
MHz = 1 million cycles/second
2. Ns or Nanosecond
(Used to measure the speed of the physical ram memory)
Nanosecond = 10^{-9} or one thousand millionth of a second
3. Ms or Millisecond
(Used to measure the speed of the hard drives)
Millisecond = 10^{-3} or one thousandth of a second

To sort of grasp what this means, when a 200MHz CPU cycles once (that is executes one instruction), this is equal to 5 Nanoseconds. The SDRAM memory used in most servers and workstations today has an average speed of 10 Nanoseconds. This means it cycles once in 10 Nanoseconds. One millisecond on a hard drive is equal to 1 million Nanoseconds. The average seek time (time it takes a hard drive to find the location of the data requested) is 8 Milliseconds for the faster hard drives to 12 or more Milliseconds for the slower hard drives. This is 8 to 12 million (yes, that's *million*) Nanoseconds. Then the hard drive has to wait for the platter that the data are on to spin under the head in order to start reading the data. This process is known as *disk latency*. The average disk latency ranges between 2.8 milliseconds for the fastest hard drives up to 6 milliseconds and sometimes more for the slowest hard drives. In Nanoseconds that would be 2.8 million to 6 million Nanoseconds and greater. You can see that it has already taken between 10.8 million to 18+ million Nanoseconds just to find the data and get ready to read it. The actual time to read the data would naturally depend on how many data are being read. During this time (18 million Nanoseconds), the CPU could have processed 3,600,000 instructions and the memory could have cycled 1,800,000 times. By now, you should have no trouble seeing which component is the most likely candidate for *bottleneck bandit* of the moment honors.

Of the three main computer components, the hard drives are the only component that actually has physical movement of machine parts when moving data. Data are moved electronically between the hard drives, memory and the CPU. Moving data electronically usually means at about the speed of light. Knowing this, it is totally understandable that the hard drive/s are going to be the slowest of the three main computer components.

Hurray!
**We've found our bottleneck and achieved the Goal of Performance
Tuning, haven't we?**

If the hard drives are the slowest components of the whole system, why don't we just find the fastest hard drives we can and put them in our computers. Then the whole system would speed up wouldn't it?

To a certain extent this is true. However, this is not always the case. What if the application you are using doesn't require a lot of data but instead computes a very large matrix in memory from a small amount of data? In this situation, the component causing the bottleneck would be the memory and/or the CPU. Or, that is to say, the bottleneck could be the memory, the CPU or both. What if on the other hand, you had a job that required extensive use of the CPU's Floating-Point Processor? This could be any job that required many math calculations. Then the CPU itself could become the bottleneck.

This could also be true when a server is acting as a file server, application server and print server all in one (as is the case in many smaller shops). In this situation, the CPU can become so overloaded that it actually begins to thrash among the many things it's trying to do. I'll talk about thrashing later. But right now, what you need to be aware of is the fact that the CPU can become so overloaded that it becomes the bottleneck.

There are also times when the memory becomes the bottleneck but the hard drive/s appear to be the culprit. In this situation, the physical ram memory has become full and the application/s are requesting more memory. Windows NT will grant these request/s for more memory by paging. Paging is a technique that the operating system (OS) can use to temporarily move data with the least activity out to a special reserved space on a hard drive. Later, when/if the data are needed again, it can be paged (moved) back into memory so that the requesting application/s can use it. We saw earlier that using memory is a lot faster than getting data from a hard drive. Now the Operating System (OS) is using hard drive space as memory. This means that our memory is now running only as fast as accessing the hard drive. You want to avoid this at all possible cost. Adding the fastest hard drives in the world would help only a little in this situation. So, is it ever a good idea to page? Well, if you don't have enough physical ram to complete a job and this is the only way to get the necessary memory, it would be infinitely faster to page than never finishing the job at all. Otherwise, avoid paging as if it were the plague that it is.

As we've seen, the hard disk is not always the bottleneck. The CPU or memory or a combination of CPU and memory can also become the bottleneck depending on what the application is doing and what resources it is using the most. So, how can you know for sure what the bottleneck on your system is? We're "on the road to find out".

Finding the Bottleneck

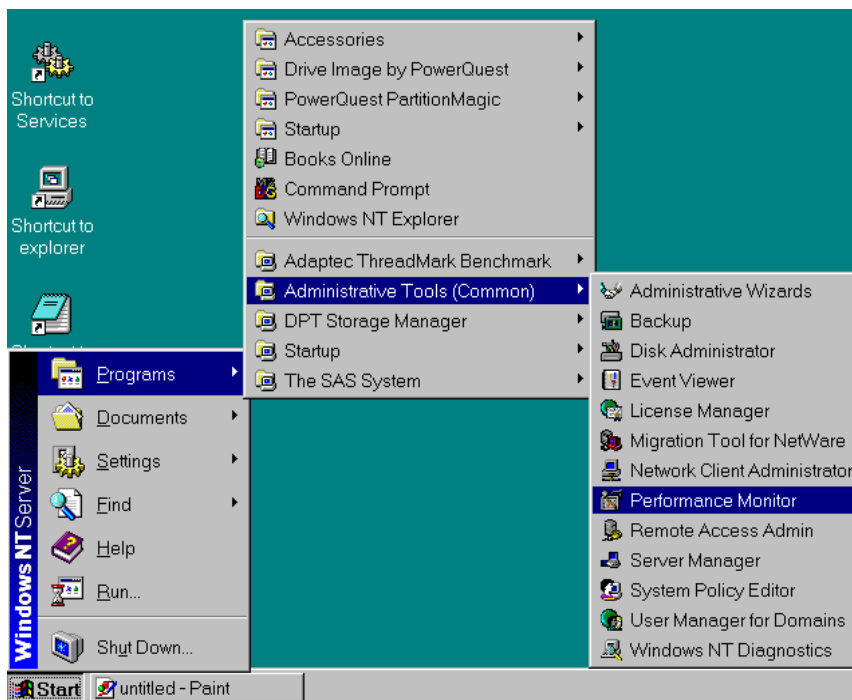
Finding the bottleneck on a system is not the easiest task, but it doesn't have to be all that hard either. Especially if you know how and that's what this section is about. I'll discuss the built in performance monitor that came free with the Windows NT operating system. And more importantly, I'll show you what to watch in order to help you find your systems bottleneck/s. But please don't consider this a full discussion on the performance monitor. There are many other ways to use it that may yield even better results. It will be in your best interest to use this as an introduction and continue to investigate the performance monitor's capabilities.

So, exactly what is the performance monitor? According to Microsoft it's "a graphical tool for measuring the performance of Windows NT computers on the network".

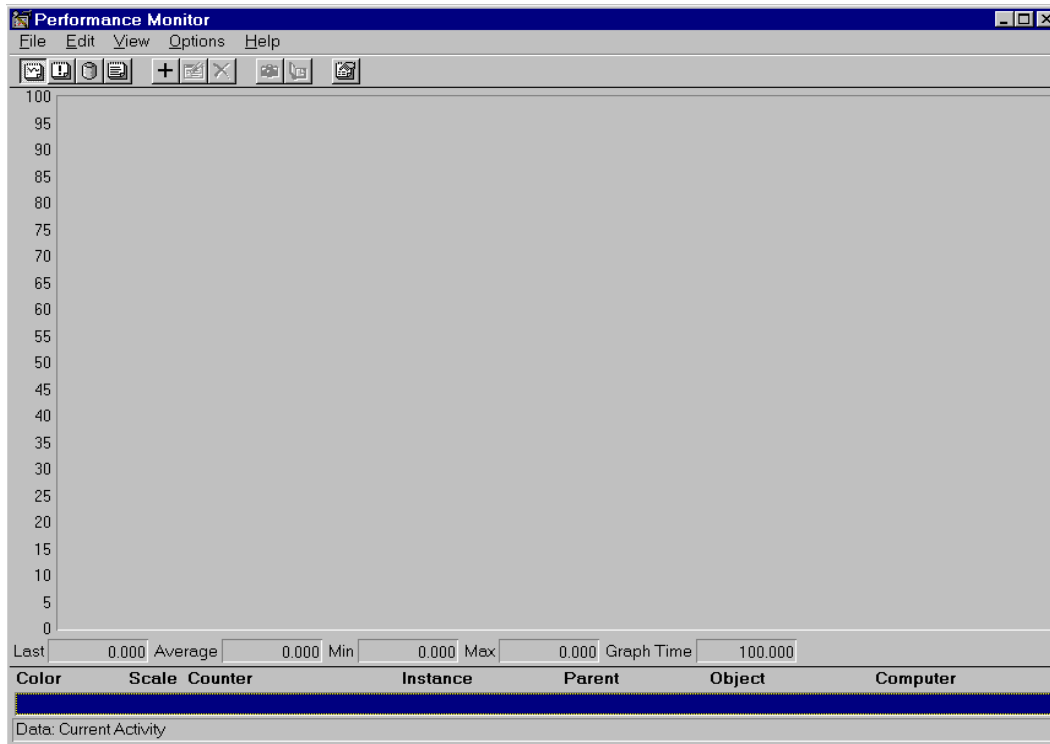
Some of the things that it can be used for are:

- a. Isolate bottlenecks.
- b. Tune your system to achieve the best performance.
- c. Show what resources an application is using and what could be adjusted to make that application perform better.

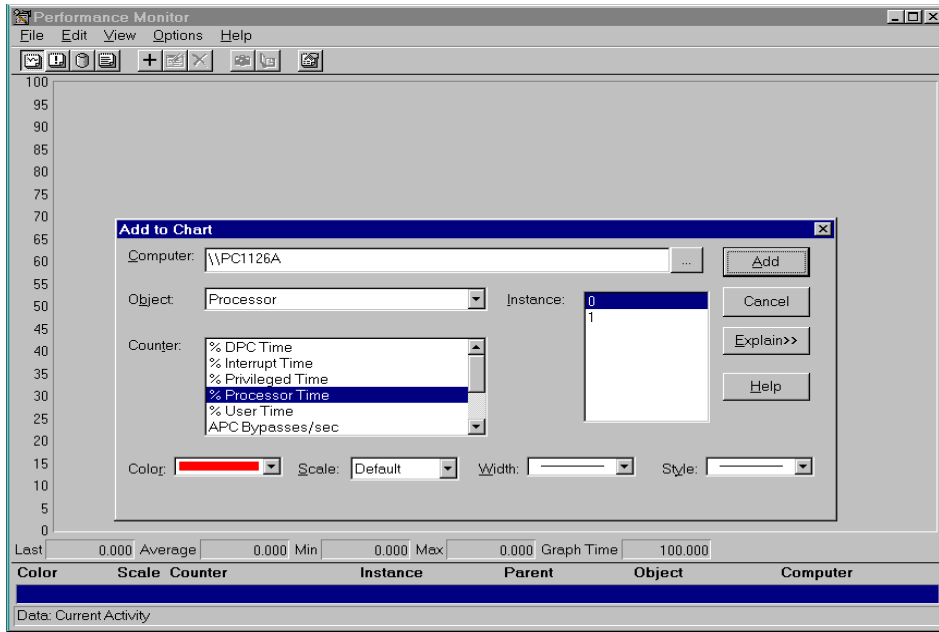
Microsoft did not design it to be the only tool you will ever need or use. Yet it can be a big help in many cases. To start the performance monitor, click the Start menu item. Then click the Programs item on the Start submenu. On the Programs submenu, click on the Administrative Tools item. You will find the Performance Monitor on the administrative Tools submenu.



After you click the Performance Monitor item, you should see something like this next example.

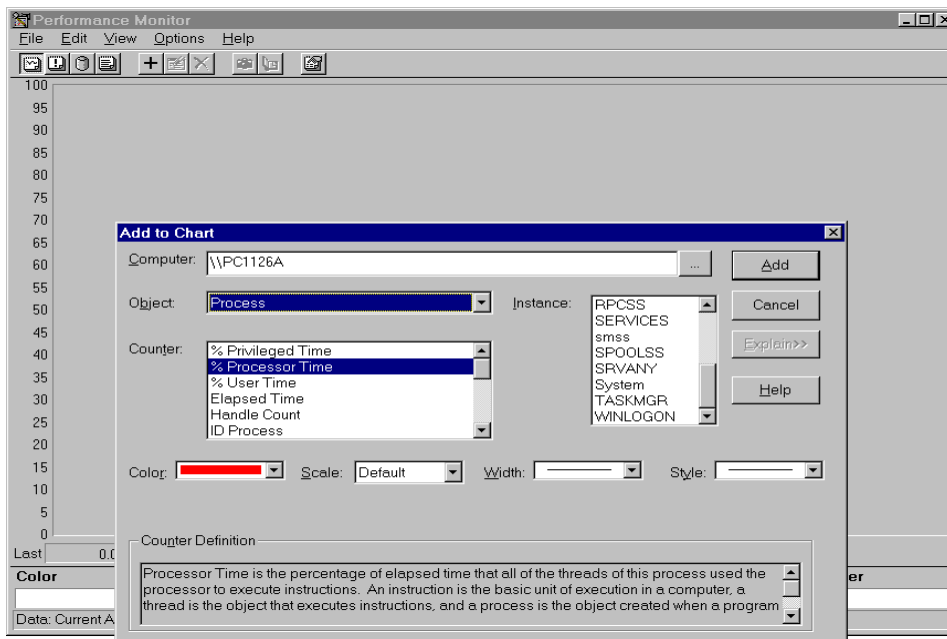


Altogether, it doesn't look like much when it's first opened. What you have to do in order to make it interesting (and useful) is select Objects and then counters for the Objects. On the left hand side is a default scale that can be sized to make it more useful and readable when using different objects/counters. The Windows standard drop down menu item is visible at the top. Right under them is a toolbar that has the following features. View a Chart, View the Alerts, View output log file status, View Report Data, Add counter, Modify selected counter, Delete selected counter, Update counter data, Place a commented bookmark into the output log, and Options. You can experiment with the toolbar items but right now we're interested in getting some objects and counters selected. There are two ways to add objects and counters. You can click on the Windows drop down menu item 'Edit' then select 'Add to Chart'. What you will see next looks like the following.



The second way is to Click the big plus sign on the toolbar, which will get you here in one click.

In the Object block, you can use the arrow to scroll through the list of Objects. Selecting an Object changes the list in the Counters block to the appropriate counters for the Object you have selected. A **big tip** here and a key to understanding the Objects and the Counters is using the 'Explain' button on the right hand side of the 'Add to Chart' dialog. The 'Add to Chart' dialog will expand to include a Counter definition block in which the explanation of the Counter can be viewed as in the example below.



The Objects and Counters

Ok, so now you know the basics of getting the performance monitor up and running. Now all you need is to know is which Objects and Counters to use right? I want to emphasize again that what I present here is only some of the many objects and counters that can be used to find bottlenecks. What I'm concentrating on is the big three stumbling blocks, which are the CPU, Memory and I/O subsystem.

Memory Objects and Counters

If asked "What is the one thing you can do to speed up my applications?" and I didn't have a tool like the Performance Monitor to use to evaluate the system, I would probably respond with "Add more memory." That's the quickest and probably the least expensive thing you can do to gain speed in most instances. However, since the Performance Monitor is included in every installation of Windows NT, my response is first look at how you have memory tuned (see appendix two). Next take a look at the memory using Performance Monitor and the following Objects and counters to determine if memory is a bottleneck to your application. One thing to remember however is that even if memory is not a bottleneck, you may be able to benefit by having extra memory for NT to use as cache. This seems to be especially true when a lot of heavy I/O is occurring. Keep in mind that a lack of memory can mask itself as slow I/O or overloaded CPU/s. The following objects and counters may be helpful when evaluating memory.

Object: **Memory**

Counter: **Available Bytes** (amount of free virtual memory).

For NT Workstation or an NT Server tuned as an Application Server, anything below 4 MB (megabytes) or an NT Server tuned as a File/Print Server anything below 1 MB, consistently would indicate that paging is occurring and performance is being degraded due to a lack of physical ram.

Object: **Memory**

Counter: **Pages/Sec** (the number of pages read from or written to the disk in order to resolve memory references to pages that were not in memory when they were referenced). If the average is consistently greater than 10, then memory is probably a bottleneck. If the average is consistently > 20, then system performance can become significantly degraded. Disk thrashing may be occurring.

Note: If Pages/Sec is increasing but Available Bytes is not going down, then you may not have a memory bottleneck. Instead, you probably have an application that is doing a lot of I/O. This is because the Pages/Sec counter goes up each time a read occurs that is not in the cache. If you're using Data Warehousing, then this counter will not provide useful information for performance tuning.

Object: Server
Counter: **Pool Non-paged Failures** (number of times something was called for but was not actually in physical memory). This is a good indicator that memory is too small.

Object: Server
Counter: **Pool Non-paged Peak** (max number of bytes of non-paged pool that has been in use at any one point). Indicates the amount of physical memory that should be on the computer.

Object: Server
Counter: **Pool Page Failures** (indication that either physical memory or paging file is near Capacity).

Object: Memory
Counter: **Committed Bytes** (Size of virtual memory that has been committed and not reserved). If it is > than physical memory, then this is a good indication that you don't have enough memory to accommodate all that the current applications are doing. However, since some paging is inevitable, it would be good to check Memory Page/Sec (mentioned earlier) and Memory Page Faults/Sec. If Memory Pages/Sec > 10 and Memory Page Faults/Sec is greater than Memory cache Faults/Sec, then too much paging is occurring.

Warning:

If your server has Pentium processor/s instead of Pentium Pro's or Pentium II's, adding memory might slow you down. Here's the scoop. Most Pentium motherboards come with COAST modules on them. COAST stands for 'Cache on a Stick' meaning cache memory. The usual amounts are 256k or 512k. The cache is used to hold instructions and data so that the CPU can gain faster access to them as opposed to having to access the main memory. The system employs a Cache Controller that anticipates what is going to be needed and moves instructions and data into the cache to have it ready for the CPU. Naturally, if the Cache Controller does a good job, the CPU finds what it needs in cache memory. Since this memory is much faster than the main memory, the CPU can access it faster. This helps keep the CPU working without having to wait for the data and/or instructions. Overall system performance increases as well as productivity. If there is a cache miss that is the data or instructions needed by the CPU are not in the cache memory, then they must be fetched from the main memory. Accessing main memory in this manner will slow the overall performance of the system. With that established, how does the CPU know if what it need is in the cache memory or whether it needs to fetch it from main memory?

On Pentium motherboards, there is a chip called a Tag Ram Chip that caches the contents of the cache memory. So, when something is needed, the CPU checks the Tag Ram to

see if what it needs is in the cache memory. If it isn't, then it fetches what it needs from the main memory. So, why are we telling you this? The Tag Ram on most Pentium motherboards will only cache 64MB of main memory. If you put more than that on your system, then the Tag Ram will not be able to tell the CPU about data or instructions that are located in addresses above 64MB. This could cause the CPU to have to search the cache memory first. If it doesn't find what it's looking for, then it would have to search the main memory. This can slow the system down so much that it has been recommended that the cache be turned off via the BIOS. This situation is usually correctable however. Most motherboards that have this limitation can be upgraded by adding more Tag Ram memory. This makes all the memory that can be placed on the motherboard cacheable by the Tag Ram. So, the lesson here is check to see whether you need to add Tag Ram whenever you upgrade the memory on Pentium based servers. Pentium Pro and Pentium II based servers do not have this problem.

I/O Objects and Counters

Once you have insured that there is enough physical ram memory to keep virtual paging down to a minimum, the next thing to look at is I/O. The SAS System has always been a heavy I/O user simply because our users generally work with a lot of data. Optimizing the speed of the I/O subsystem can usually do more for our users than anything else (provided enough memory is available). In order to evaluate I/O, the following objects and counters may be helpful. But first you will need to enable the disk-monitoring capabilities by enabling the disk counters. These counters are turned off by default since they add an overhead of about 1%. The Diskperf utility is used to start these counters. All you have to do is type Diskperf -y at a command prompt. If you're not sure whether or not Diskperf is started, just type Diskperf at a command prompt and the system will tell you whether the disk counters are enabled or not. You must start the counters before you start Performance Monitor. Be sure to stop the disk counters after you've finished monitoring the hard drive/s so that the 1% overhead doesn't continue to hurt your systems performance. To do this just type Diskperf -n at a command prompt.

Object: **Physical Disk**

Counter: **%Disk Time**

If disk queue length > 2 and %Disk Time is consistently high (=>67%), there is a good possibility that the hard drive is a bottleneck.

Object: **Physical Disk**

Counter: **Average Disk sec/Transfer** (average disk transfer time in seconds)

If the controller is retrying the disk continually because of failures, then this counter will have a high value (> 0.3 seconds).

Object: **Physical Disk**

Counter: **Disk Queue Length** (* pending disk I/O request).

Disk congestion could be assumed if this counter is > 2.

*Note: The number of requests outstanding on a disk.

Object: **Physical Disk**
Counter: **Disk Bytes/sec**

If the count is lower than 20k, this is an indication that an application is accessing the disk inefficiently.

Another performance measurement that can provide a way to compare how a hard disk is performing relative to other hard disk at your site is the Average Queue Time. This is the time that it takes for a disk transfer (including both reads and writes). To get this measurement you will need to use the following formula:

$$\text{Avg. Queue Time} = \text{Disk Queue Length} * \text{Avg. Disk sec/Transfer}$$

CPU Objects and Counters

The SAS System **does not** take advantage of multiple processors in a system. That is to say, **the SAS System is not SMP compliant**. If you have multiple processors, and start a second SAS job, NT can decide to assign the second CPU to the second SAS job. In this type situation, multiple CPU's would be advantageous. For a server environment where lots of other things are happening as well as applications being run, multiple processors may be a necessity. But don't count on adding CPU's to increase the speed of a particular SAS job. It just won't happen unless of course the added CPU's relieve the load on the single CPU without causing a lot of resource contention. The following objects and counters might be helpful in determining if your system could benefit from adding more CPU's.

Object: **Memory**

Counter: **%Processor Time** (amount of time the processor is in use).

If consistently > 80% and disk and network counter values are low, adding a faster processors may help.

Object: **System**

Counter: **%Processor Time** (this one is for multi-processor systems).

If consistently > 80% and disk and network counter values are low, adding faster processors or more processors may help.

Object: **System**

Counter: **Processor Queue Length**

If > 2 for a sustained period of time, processor/s could be a bottleneck.

Object: **Processor**

Counter: **Interrupts/Sec**

If this counter increases dramatically without a corresponding increase in system activity, a system hardware problem is apparent.

Object: Processor (_Total)
Counter: %Processor Time

The need for additional processor/s may be indicated if several processes are in contention for most of the processor/s time.

In general, what we think you're more likely to find is an I/O problem. There are several reasons for this.

1. Hard drives don't make development gains at the same rate as memory or CPU's.
2. Most SAS users are working with lots of data.

Let's take a look at the physical characteristics of the I/O subsystem in order to get a better understanding of why they are likely to be a bottleneck in most systems.

I/O Subsystems

There are several components that make up the physical aspects of the I/O Subsystem. They include but are not limited to:

1. Physical ram used as Cache memory
2. I/O bus
3. Controllers for the hard drives
4. Hard drives

What's this? Cache memory is a component of the I/O subsystem? Many of you know what a cache is hopefully. For those of you who don't, it's just a place in memory that Windows NT uses to store data that it thinks will be needed in the near future. There are other kinds of caches too, like the cache built into a hard drive or cache that is built into some of the controller interface cards that we'll talk about shortly.

Some of the testing we've done indicates that not only is cache a part of the whole scheme in terms of I/O but it is also the fastest component of the I/O subsystem. Having a surplus of memory to cache files can significantly increase performance for some types of applications. How is this possible? Here's the scoop. Windows NT uses any memory that is not in use as a dynamic cacheable area. What this means is, NT can increase the size of the cacheable area as memory becomes available and it sees a need to cache data.

It can also shrink the size of the cacheable area when/if more memory is needed by applications that are running. But let's suppose that we have a lot more memory than the currently running application needs, and one of those applications is accessing a large file. In a situation like this, we have seen Windows NT use this large cacheable area to cache the file that is being accessed. It actually pulled the whole file into memory anticipating that the application using the file would need it. By doing this, the application did not have to wait for reads to occur from the hard drive since the data were already in memory. This provided performance that we had not been able to achieve before. We'll discuss our testing in depth later. Right now let's have a look at the rest of the I/O Subsystem.

Next in order of speed is the I/O bus itself. Most Pentium computers today use the PCI bus for I/O. This bus has a maximum transfer rate of 132mb/sec. To compute the maximum transfer rate of the PCI bus multiply the bus width by the cycle time. The Bus width is 32 bits or a 4-byte word. The cycle time is 33ns. Multiplying these together gives us a 132mb/sec possible transfer rate. Since the best sustained transfer rates for hard drives cannot even come close to this figure, we don't believe that the I/O bus presents an obstacle to performance on most of the smaller systems. On larger NT servers that are setup as a file server, and that have lots of I/O devices such as RAID packs, CD-ROM towers, tape systems... Etc., then the 132mb/sec bandwidth may become a bottleneck. The new 2.1 PCI specification addresses this by doubling the speed of the PCI bus to 66ns. This effectively doubles the amount of throughput for the I/O bus to 264mb/sec. But we'll all have to wait since this new specification is not being implemented just yet.

Each of the two major hard drive types (EIDE and SCSI) has a controller built into it. The controller manages request for data and acts as an interface between the I/O bus and the hard drive.

On an EIDE hard drive the controller interfaces directly with the chipset of the computer. This allows EIDE hard drives to be plugged directly into built-in IDE ports or channels. IDE slots built into the motherboard utilize the PCI bus only and almost all motherboards on the market today have at least two IDE channels. Having these channels already available helps to minimize the cost of adding more IDE or EIDE hard drives. This is because you don't have to add a more expensive controller interface card like you used to with the older MFM or RLL hard drives. On the downside of this, the motherboard has only two channels, and each channel can handle only two devices. That limits the IDE devices to 4 per motherboard unless a tertiary IDE channel is added. Installing an IDE interface card in an ISA or PCI slot or utilizing the IDE channel built into some sound cards are two ways to add an extra IDE channel. IDE interface cards are less expensive than SCSI interface cards because they do not have to have controller circuitry built in.

With SCSI hard drives it's not as simple. Each SCSI device has a controller built into it. This is also true of the SCSI controller interface card that must be installed in order to use SCSI devices on a computer. A SCSI device's built-in controller 'talks' to the controller on the SCSI controller interface card in order to transfer data. So, unlike IDE drives, you must install a more expensive SCSI controller interface card in order to utilize SCSI

devices on a computer system. This adds to the cost since these interface cards are more expensive than the IDE channels already existing on most motherboards.

You can buy SCSI interface (controller) cards that plug into either an ISA or a PCI slot. SCSI controllers are also starting to show up on the motherboard. An advantage to SCSI is once you've added a SCSI controller card, you can chain a number of SCSI devices onto it. The number of devices depends on the SCSI controller card you install. Narrow controller cards will handle only 8 devices with one device being the controller card itself. A wide controller card will handle 15 devices plus itself. Our testing did not show the controller to be a bottleneck in performance. Instead, what we found in most cases was that the hard drives were unable to provide more data than the controller could pass along even when using striped sets of four drives (striping will be explained a little later when we get into RAID hard drive systems).

So, that leaves us with the last main component in the I/O Subsystem. The Hard Drive. Technological advancement of hard drives has not kept pace with memory, CPU's or the other electronic components in a computer system. Please don't misunderstand. Hard drives are getting faster. They just aren't making the tremendous gains in performance that the other computer components are making. This has opened the door for lots of companies to come up with new and interesting ways of integrating hard drives together in order to seek higher I/O performance levels. Have you ever heard the term 'RAID' in relation to computers? RAID stands for Redundant Array of Integrated Drives. Sometimes we hear RAID referred to as Redundant Array of Inexpensive Drives. This is an older term. If you take the time to investigate the cost of the hard drives used in some RAID systems, you'll know why the former is probably closer to being a better fit for the proper definition of RAID.

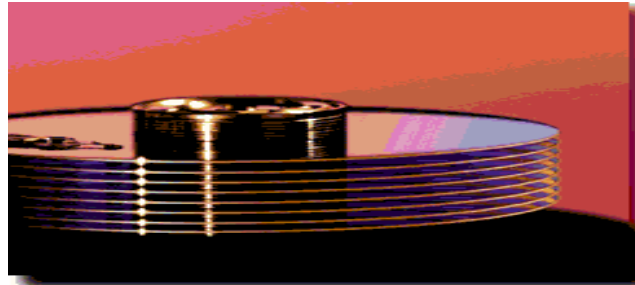
Either way, RAID describes a way of using several hard drives working together to improve I/O performance. Let's take a look at how a hard drive works so we can understand why the slowest component in a system is usually the hard drive. Along the way, we'll also try to understand RAID systems to get a firmer grip on what they are and how they help speed the I/O process up. This discussion will not be a complete tutorial on RAID systems. Please consider it only as an introduction to RAID.

Section II

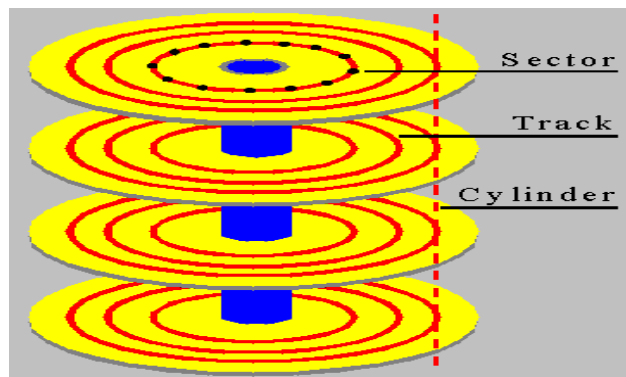
The Hard Drive

Have you ever seen a Hard Drive? Most of them are these small little rectangular boxes about 4 inches wide, 5 $\frac{3}{4}$ inches long and 1 inch thick. Some of them are chrome while others are black and others are just dull looking aluminum with maybe some black on the sides. And if you turn them over, there is usually a circuit board with all these little electronic parts sticking up here and there. It's amazing that something this small could hold so much data. And every year that goes by, the hard drive manufacturing companies find new ways to pack the data together tighter and tighter on the hard drives little platters. This translates into the platters that can hold more and more data. Without having to add more platters to hold more data, the price of the hard drives keeps getting smaller and smaller. And that pleases all of us. But wait, what is a platter?

Glad you asked. Take a look at this picture and you will be able to see a whole bunch of platters.



Each round disk is called a platter. On the left side mid way down and on top of the first platter is an arm sticking out with a little bump on the end. This bump is the head that actually reads the data from the platter. Each platter has a head that can move in and out as needed to find the data that has been requested. All heads move in and out as a single unit though. That is to say, the heads do not move independently. This is an important point that we will explore shortly but first we need to examine the make-up of a hard disk in more detail.



In the above picture you can see that each platter of the hard disk is divided into tracks and sectors. Each sector can hold 512 bytes. Different size hard drives have different numbers of sectors per track. This relates to the density of the media on the platter that holds the actual bits. The tighter the bits can be stored together, the more sectors you can have per track. More sectors per track means more data per track. This in turn means more data per platter, which means more data per hard drive. Another way to increase the amount of data on a hard drive is to add platters, but remember the width of a typical hard drive that was mentioned above. With only one inch to play with, adding platters is not very easy. To a lesser degree, but still another benefit of denser media is that more tracks can be added to each platter. More tracks also mean more data per platter and more data per hard drive. Now you know why the density of the media is so important to hard drive manufacturers as they increase the size and speed of their offerings.

Did I mention speed as a benefit of denser recording media? Oh yeah, I sure did. Well, it works like this. In order to read the data you have to get it under the head. This may sound stupid, and you probably already know this, but the only way to do this is to make the platters spin round and round. Faster rpm's mean more data traveling under the read head in a given time.

It's partially the speed at which the platter's spin (known as spindle speed) that the transfer rates are based on. Sorry, there's another one of those terms that needs defining in order to be able to get the message across, so, bear with me here. The transfer rate is the rate at which the data can be read from the hard disk. Mostly we're interested in a 'sustained transfer rate', which means the rate at which data can be read from the hard drive on a continual basis. The sustained transfer rate can be computed as follows:

Sustained transfer rate = # of bits/sector * # of sectors/track * (the rpm of the hard drive divided by 60)

For example, a sector is 512 bytes. Multiply this by lets say 164 sectors per track times a 7200 rpm hard drive divided by 60 and we have a sustained transfer rate of 10mb/sec. Why do we divide the rpm by 60? Well, rpm means revolutions per minute and we want to get the number of megabytes per second. Sixty seconds in a minute...., you get the picture right? One little warning here about computing the maximum sustained transfer rate. This is a theoretical limit and not always a realistic value.

There are also terms like 'burst rate' associated with reading data from the hard drive. What burst rate means is the amount of data that can be read from the hard drive in a given moment but can't be sustained over a longer period. When purchasing hard drives you don't want to base your decision on burst rates which the salesman will probably throw at you. Always inquire about the maximum sustained transfer rate. This will more than likely send them scrambling for the specification sheets. You also don't want to base your buying decision on the average seek time either (also known as average access time). Almost everyone uses this as his or her major selling point. Yet, it can't tell you a thing about how much data a hard drive can transfer. We'll jump into this next. First let me finish what I started on speed and denser media.

You now know what the maximum transfer rate is and how it is computed. You can easily see that the speed at which the data moves under the head is a major factor. What about the amount of data that can be on one track though? If you multiply the 164 sectors that we used in the example above by 512 bytes (number of bytes in a sector), then we come up with 83,968 bytes per track. Since you can only read one track at a time, and since the rpm is fixed, if we put more data on that one track, then more data would travel under the head in the same amount of time. We would therefore increase the amount of data that could be read in a given amount of time.

Example: if we increased the density so that the number of sectors per track is doubled (328 sectors instead of 164 as in the example above) using the same rpm and 512 bytes per sector, we now have a sustained transfer rate of 20mb/sec.

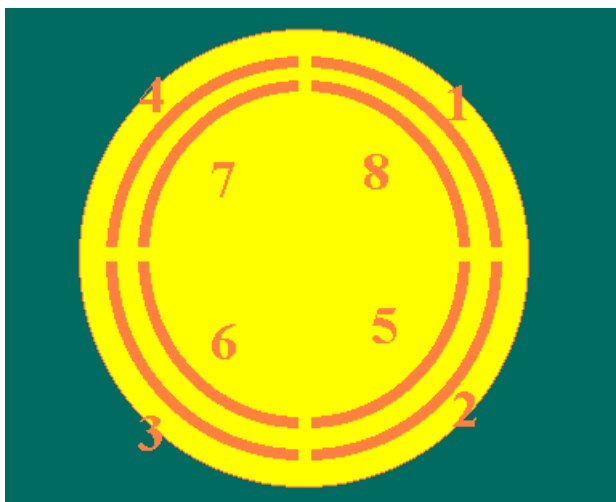
$328 * 512 = 167,936 * 120(\text{rpm of } 7200/60) = 20,152,320 \text{ bytes/second or } 20\text{mb/sec.}$

Doubling the number of sectors per track has clearly doubled the sustained transfer rate while keeping all other factors the same. Hopefully, this makes it very clear why the density of the hard disk media is so important to manufactures and to consumers.

Enough said on density, let's discuss average seek time. Average seek time is the average time that it takes to move the head to the track that the data can be found on. This is important if your application/s is/are doing a lot of random accessing of the data. If on the other hand you are reading files sequentially, the larger they are the less important average seek time becomes. This is because once you find the data, the hard drive will start reading the data and will not be seeking until the file has been read. Depending on the size of the file, this could take a while.

One last important term and we'll wind up our discussion about the internal workings of the Hard Drive and get into the different types Hard Drives. Once the hard drive has found the track that the requested data are on, the data may not be in the sector that is currently under the read head. When this happens (which is usually the case) the hard drive has to wait until the platter spins the data under the head before it can start reading. This is referred to as rotational latency. Rotational latency is usually encountered whenever the hard disk has to move the read head to a different track. This is an important point that needs to be understood in order to keep your system at its optimum performance once you've gotten it tuned properly.

When randomly accessing data, average seek time can really start mounting up. With sequential access, the data are usually written on contiguous sectors and then when the track is full the next sector of another track is used to continue the file. If all the data of a file are written to contiguous sectors on contiguous tracks, then the data will be in the best possible situation to be accessed as fast as possible. This keeps the average seek time down to finding the beginning of the file. And it also keeps the rotational latency down too. To understand this, take a look at the following:



As you can see, sector 5 on track 2 is positioned so that the head has time to move to the next track and be ready to continue to read the data. When accessing a large sequential file, this would help to keep average seek time as well as rotational latency to a minimum. However, over time disk drives become fragmented. What this means is, over a given period, as different size files are written to disk then deleted, the sectors to hold data may not be contiguous when a file needs to be written to disk. You see, when the OS writes data to a hard drive, it looks in the Master File Table (also known as MFT) to find the starting point for the file. The MFT is sort of like a table of contents for your hard drive. Each sector contains a pointer to the next sector so, data do not have to be in the next sector in order to be found. Instead, the next sector could be anywhere on the hard drive. This could cause a lot of head movement and lost time with rotational latency if the disk is very badly fragmented. But as most of us know, disk fragmentation occurs as time goes by and sequential data can be written to the hard disk so that it is 'all over the place'. If this happens, the excessive head movement as well as the rotational latency will cause the performance to drop sometimes drastically. The moral of this little story is 'You must run a program to defragment your drive occasionally. We have heard claims that Windows NT's NTFS file system doesn't ever have to be defragmented. This is not entirely true and it is in your interest to find and use a good defragmenting program on a continual basis.

Rotational latency can be computed for a hard drive as follows:

$$\text{Average rotational latency} = 1 / \text{RPS (revolutions per second)} / 2$$

Example: A 7200-RPM hard drive spins at 120 revolutions per second. Divide 1 by 120 and you get 0.008333333333.... Divide that by 2 and you get 0.004166666666.... or 4.17msec. This is the rotational latency of the Seagate Fast and Wide drives that we used in our testing.

'There I go again', spouting off terms that I haven't explained yet. Fast and wide is a term used to describe a type of SCSI hard drive. But wait, what is SCSI? Does everyone know what SCSI means? The term SCSI refers to a specification known as Small Computer System Interface. It has been around for a long time. In the early days of personal computers, there were many problems with SCSI devices. Installation was a nightmare and there didn't seem to be any agreement on what the true specification meant. It's a wonder that it didn't die out. Eventually, progress was made on determining what the specification should be and SCSI came to be known as the speedier way to go if you wanted fast performance from a hard drive. Today, SCSI drives have corralled the title of fastest hard drives and are almost exclusively used in the RAID systems that we'll talk about in a minute.

Here is a list of the different SCSI standards in use today and what they mean. This may not mean a lot to some of you, but what you can get out of it is the fact that there are many types of SCSI being implemented. All these different terms associated with SCSI makes it harder to understand just what's right for your needs. Remember though, 'knowledge is power' and in this case understanding SCSI means performance.

SCSI-2

ANSI standard X3.131-1994. Included 1 byte wide data bus, defines FAST transfer speeds, defines SCSI protocol for wider data transfers, and defines parallel SCSI messages and command structure. Easier installation was a benefit of SCSI-2.

SCSI-3

The ANSI specs for SCSI-2 is very large. SCSI-3 splits it into a series of smaller documents. Each document covers a different layer of the SCSI-2 specification so that as new SCSI technology emerges, substitution of parts of the structure are allowed for.

SCSI FAST

Refers to timings defined in SCSI-2. Transfer rate is defined as 10mb/sec.

SCSI FAST-20

This refers to timings defined in SCSI-3. Transfer rate is defined as 20mb/sec.

SCSI FAST-40

Timings are for future revision of SCSI-3. Transfer rate is defined as 40mb/sec.

Ultra SCSI

An Old term that describes the FAST-20 data rate. It has now been dropped.

SCSI WIDE

A two byte wide connector (68 pin) defined in the SCSI-3 Parallel Interface document.

SCSI FAST-WIDE

This is a combination of a two byte wide connector and a FAST transfer rate. Transfer rate for wide FAST-20 would be 40mb/sec and 80mb/sec for wide FAST-40.

Fiber Channel SCSI

Any product that combines “fiber channel physical and protocol characteristics with SCSI command set”. Basically, it’s a serial interface that uses a 1 GHz (gigahertz) signal to achieve a 100mb/sec-transfer rate over a coaxial cable.

Whew! Lot’s of SCSI stuff to talk about isn’t there? IDE is almost as bad. Ok! Ok! I know. I know. What is IDE? IDE is another hard drive technology and a competitor to SCSI. As a matter of fact, if you have a computer on your desk, it probably has an IDE hard drive in it. This is mainly due to cost. SCSI hard drives are faster but they’re also more expensive. Therefore, SCSI hard drives are usually found on high performance computers and servers.

IDE is defined as ‘Integrated Drive Electronics’ and is used by most of the industry with the exception of Seagate. Seagate likes to refer to this interface as ATA (Advanced Technology Attachment). It generally means the same thing. IDE evolved into EIDE or ATA-2 and finally into Ultra DMA/33 or Ultra-ATA. The IDE standard depends on the motherboard having the ability to act as a controller and the hard drive to also have a controller built into it. The motherboard gets its controller ability through the chipset on the computer. The chipset, such as the Intel TX chipset, defines and provides the capabilities of the motherboard. So, in the same way that SCSI uses two controllers in order to perform I/O, IDE basically does the same thing.

If IDE drives are not usually used on NT servers, why do we need to know anything about them? The simple answer is, there has been a lot of advancement in the IDE standard. While the hard drives in this class can’t match the fastest SCSI hard drives, they’re starting to give the Fast and Wide SCSI hard drives a run for their money. Until now, the fastest EIDE hard drives ran with a spindle speed of 5400-RPM’s. The newest offerings from Seagate now run at a 7200-RPM spindle speed. This is a first for EIDE drives and matches the spindle speed of the Fast and Wide SCSI hard drives. But I’m getting ahead of the story here.

In the beginning, IDE hard drives had an initial speed of 3.3Mb/sec in PIO mode 0 and 2.1Mb/sec in DMA mode 0. PIO mode was supported by all PC devices and worked something like this. When the OS executed a read or a write, the data was moved one, two, or four bytes at a time to the device. The CPU was directly involved with each transfer. This worked well for simple devices. DMA (Direct Memory Access) mode was different. A smart device could transfer data directly to memory without the use of the CPU. This was made possible by a set of DMA controller chips on the motherboard that could generate the buffer addresses in memory instead of having to rely on the CPU. Each chip had a level number and could only support one device. This was supposed to free the CPU so that it could continue to run applications. The problem with the early adaptation of DMA was that when a device was ready for more data, one bus cycle was

used to send a request to the DMA chip. This one extra bus cycle was one more than PIO mode used and therefore made DMA slower.

When the PCI bus became available, things changed for DMA mode. Every adapter card had access to a 32-bit address bus. This allowed adapters to generate addresses that could reach any location in physical memory. This meant that adapter boards could have their own DMA controller chips. They now had the ability to generate the sequence of memory addresses that would allow them to read and write data between the memory and the adapter. Bottom line was data could be moved in every I/O cycle on the PCI bus. This technique is known as bus mastering.

There are presently 4 PCI modes and 7 DMA modes. Most notably are PIO mode 4 or multiword DMA2 that achieves 16.6Mb/sec (known as EIDE) and multiword DMA 3 that allows up to 33.3Mb/sec data transfers. Multiword DMA 3 is referred to as Ultra DMA/33 or Ultra-ATA. Ultra DMA/33 class hard drives are for the first time presenting a real challenge to SCSI hard drives. While they have not completely closed the speed gap, they are far less expensive than their SCSI cousins.

That wraps up our discussion of hard drives for the moment. There are other factors that come into play such as data access time (combination of seek time, head switch time, and latency), head switch time and cylinder switch time that we are not going to go into. We've covered the most important factors and the two major types of hard drives. This will allow us to continue with our discussion on performance. Next we'll cover RAID and see what it means.

RAID

As mentioned earlier, RAID stands for Redundant Array of Independent Drives. The intentions of the early developers were to use inexpensive hard drives to provide a system that could be redundant. This was changed later to independent since some of the hard drives you can buy now are not so inexpensive. In the early days, there were six proposed RAID levels. Several levels of RAID have been combined and at least two new levels have emerged. Chances are RAID research will continue to develop new RAID levels for years to come. RAID systems come in many types, sizes and price ranges. They can keep your data safe and they can provide solid performance. And, of course it goes without saying that if you want performance, you'll pay for it if you buy a quality RAID system.

OK, so what does redundant mean? If we turn to Webster's Ninth again, we find:

'3: serving as a duplicate for preventing failure of an entire system (such as a spacecraft) upon failure of a single component'.

A computer is certainly not a spacecraft. Although, at the new speeds that these things are running, and the sounds that emanate from them, sometimes you may think it's ready to blast off. If we choose just one word in the definition that would best describe redundant when used in relation to RAID it would be duplicate. Because that's just what RAID systems do for the most part. Why are we talking about them in a paper on performance then? We're supposed to be talking speed here aren't we? Sure, but if you lose your data then it doesn't matter how fast that system used to be. It all of a sudden becomes the slowest system you ever owned. What really matters these days is to try to have as much speed as possible yet find a way to keep your data safe. And that's what RAID systems try to do except for one of the RAID levels that is. RAID 0 is meant for speed only. It does not address redundancy at all and is subject to debate on whether it is a RAID level at all. RAID 0 has been successfully combined with RAID 1 to take advantage of both speed and redundancy. At this point there are lots of questions. So, maybe we'd better get started answering them by taking a look at what kinds of RAID levels there are and what each one does.

RAID Definitions

What Levels are there and what should you use?

Disk Array - two or more disk drives connected together in such a way as to make it appear that they are one drive.

Software RAID - RAID levels that are provided by the Operating System. It is usually slower than Hardware RAID.

Hardware RAID - Specially built controllers that provide RAID levels through hardware. This is usually faster than the Software RAID that is provided by an Operating System. RAID 0 (better known as striping). Minimum of two drives. Data is interleaved between the drives so that the workload is balanced among all drives in the striped array. Very good performance but there are no provisions for redundancy. If one drive fails, all data is lost. This type of RAID should only be used when you need high-speed storage and aren't worried about redundancy. For example: a RAID 0 drive can be set up to provide storage for NT's use as virtual memory or for other temporary files such as the SASWORK subdirectory. If the RAID 0 array fails, you aren't worried about the data in it since the data are temporary. That's why it can be used to provide fast storage for temporary type data.

RAID level 1 (better known as mirroring). Minimum of two drives. I/O writes are written to identical twin disk drives (the twin being the mirror). If one disk fails, the other has the exact same data on it. There are several disadvantages. Every write has to be done twice. So, writing long files can degrade system performance. One of the ways

the industry has tried to work around this is to employ a technique known as duplexing. Duplexing can be implemented with a two channel RAID controller or two separate RAID controllers when using NT's ability to do software RAID. Each mirrored pair is placed on a separate channel or on separate RAID controllers. This basically allows writes to occur almost simultaneously and this speeds up the dual writes. The second problem is you effectively lose half the disk space that you purchased in order to provide the backup. Another way of looking at it is disk space cost twice as much. This type of RAID is best used when redundancy is an absolute must yet fast small random writes are mainly what the system is doing. Best used in small file server systems.

RAID level 0/1 - This is a combination of RAID levels 0 and 1. Minimum of four drives. Each mirrored pair is setup as a RAID 0 array. That is to say, you have a mirrored pair of RAID 0 arrays. You get the benefit of striping from RAID 0 for performance and the redundancy of RAID 1. Long writes would still affect this type of RAID. Duplexing could be used to increase the write performance. This RAID type is sometimes referred to as RAID 10.

RAID level 2 - Intended for non-error detecting drives, it uses error correction codes. Most SCSI drives support built-in error detection negating the need for RAID 2.

RAID level 3 - Uses bytes level striping across several disk drives and stores parity on one disk drive. The parity can be used to rebuild a failed drive. Is very similar to level 4. Best used on small file server.

RAID level 4 - Uses block level striping across several disk drives and stores parity on one disk drive. The parity can be used to rebuild a failed drive. Read performance is better than write performance due to having to write the parity information associated with each write. All parity data are stored on one drive. Good RAID level for larger file servers.

RAID level 5 - Similar to RAID level 4 but distributes the parity data among all the drives in the array which can slow reads slightly but speeds the writes up considerably. This is a good RAID level when reading small files or small parts of files such as a Database server or if the I/O is mostly random. If one drive fails, it can be rebuilt from the parity data.

RAID levels 6 and 7 - Similar to 5 but can recover from 2 or 3 drive failures respectively.

RAID Array Implementation

RAID levels are implemented by software code. What distinguishes the different ways RAID can be implemented is where the software is actually executed. There are basically 3 ways that RAID is implemented:

1. The operating system (known as Software-Based RAID).
2. Bus-Based Array Adapters/Controllers
3. External Array Controllers (also known as “Bridge Controllers”)

Software-Based RAID arrays are implemented through the operating system usually on entry-level servers. Windows NT for workstations provides Software-Based RAID 0, while Windows NT Server provides levels 0, 1 and 5. The RAID code is executed by the systems CPU and the algorithms can be very mathematically intensive. This can slow overall performance since CPU utilization and interrupts will increase in order to perform the RAID operations. Use this information as a guide when deciding what kind of CPU to put in an entry-level server that will implement Software-Based RAID levels. Only CPU's with the fastest FPU's (floating-point Unit, a part of the CPU that does the math calculations) should be considered in order to obtain the best performance you can get when implementing Software-Based RAID. Another hint, you usually want more than one processor on your server and there is only one brand of processor that you can do this with. This brand also can't be beat when it comes to math calculations. Software-Based RAID is considered to be the slowest way to implement RAID levels.

Bus-Based RAID Arrays are implemented by an adapter/controller that fits inside the server case in one of the motherboard's PCI slots. They generally have one or more secondary processors that perform most of the RAID operations and I/O commands. This helps to increase I/O performance as well as overall system performance because this implementation of RAID does not depend on RAID algorithms being processed by the CPU. Some Bus-Base RAID controllers also have cache built on them which is also reported to further increase the I/O performance. It has been said that this type of RAID implementation is the fastest since the RAID controller resides directly on the PCI bus. Because of cost, these controllers used to be found only on the mid-range to high-end servers. But due to continued research, development and manufacturing cost reductions, they have become available to the entry-level servers.

External Array Controllers are usually located in the external RAID enclosure and hooks into the system through one or more device channels such as a SCSI port. They also have high performance microprocessors right on board and are able to handle the execution of all RAID software code functions without interfering with the operating system. These high performance microprocessors and advanced cache algorithms help many External Array controllers be second in performance only to the Bus-Based RAID controllers.

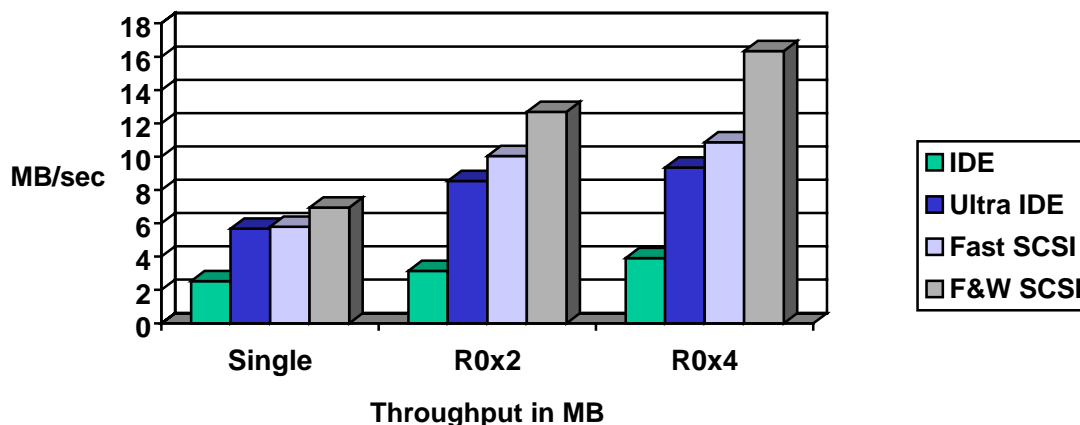
However, bets are that as Fibre Channel develops and becomes more readily accessible, external Arrays will match and maybe even surpass their Bus-Based counterparts.

RAID Discussion

All RAID levels were not tested for the next part of the paper since performance was the main focus. RAID levels 0, 0/1, 1 and 5 were tested. What follows is a short discussion about these different RAID levels based on my observations during the testing.

RAID level 0: This specification was meant for speed only. There are no considerations for backing data up unless you take it upon yourself to add something like a tape backup system. If you've ever heard the term 'striping' in relation to data storage, then you have heard of RAID 0 but maybe didn't actually know it. What it means is, there are two or more hard drives that have been combined in such a way that they can work together (in parallel) to increase the speed at which you can read or write data from/to them. The claim is that every time you add another drive to a RAID 0 array, the speed increases by 50%. This was not found to be true and it wasn't my purpose to prove or disprove this claim. How much speed each drive will add depends on a lot more things that I won't take time to address right now. Suffice it to say that I did see a nice increase in performance when I added a hard drive to the RAID 0 arrays of Ultra IDE drives but only a modest increase when I added drives to the SCSI arrays (see the graph below titled Sort Times).

When I ran the Adaptec Threadmark 2.0 test, which measures throughput and CPU utilization, it was a whole different story. The Fast and Wide SCSI drives improved the most while the Standard IDE drives improved the least (see the graph below titled Throughput in MB). This indicates that I should have seen a big performance gain when I added more hard drives to the Fast and Wide SCSI RAID 0 array. But I didn't.



Putting this into perspective I concluded that while standard test like Threadmark can tell you how fast the hard drives will perform based on standardized test, this is not an accurate prediction of how fast your applications will work on the hard drives in the real world.

RAID levels 1 and 0/1: I did not do a lot of testing here. I only tested enough to verify that RAID level 1 was in fact slower and that RAID level 0/1 was faster than RAID level 1.

RAID level 5: When I tested RAID level 5, I found it to be so much slower (44%) that I stopped right there and didn't test it anymore. (Note: These same tests may vary if tested at your site). I was using an Adaptec AAA131 RAID controller and 4 fast SCSI hard drives to test RAID levels 0, 0/1, 1 and 5. Adjustments were made to try to tune the controller at RAID level 5 but they didn't seem to make a lot of difference.

The purpose was to find performance. I concluded that I needed to test in RAID level 0 since I had established that it was indeed the fastest of the RAID levels. What I was hoping to find was something that would allow our users to increase the performance on their systems without having to spend a lot more money on them. I did indeed find several things.

Section III

The Test

If you've read all the above, you're probably starting to get a little impatient and want to get to the bottom line. In order to understand how I got to my results though, it is important to know what I tested and what I tested with. Keep in mind that you may not be able to achieve the same results. I tried these same tests on two almost identical machines. One was older than the other and had been upgraded from dual Pentium 180's to dual Pentium 200's. While the results were consistent within each machine from one hardware combination to the next, the older machine was slower than the newer one for the same hardware combinations. The lesson here is: different machines and different equipment equals different speeds.

Test Machine and I/O hardware

- 2 Dell Optiplex GXPro's
- 2 – 200 MHz Pentium Pro processors each with 256k cache
- 2 – 200 MHz Pentium Pro processors each with 512k cache
- 4 - 168 pin DIMM memory modules of 32k each
- 4 - 168 pin DIMM memory modules of 128k each
- Promise FastTrak IDE RAID controller
- Adaptec AAA-131 RAID controller with 1 MB error correcting cache (ECC) memory
- 4 Western Digital EIDE 1.2 gig Caviar hard drives (21200)
- 4 Western Digital Ultra EIDE 4.3 gig Caviar hard drives (34300)
- 3 Seagate Barracuda Fast SCSI 2.0 gig hard drives (ST32272N)
- 1 Seagate Barracuda Fast SCSI 2.0 gig hard drives (ST32171N)
- 4 Seagate Barracuda Fast and Wide SCSI 4.5 gig hard drives (ST34572W)

What was tested

SAS user's process huge amounts of data which requires lots of I/O. By now you should be aware that the I/O is the bottleneck for most SAS customers. So, I reasoned that a Proc Sort would be a good test since it would put a strain on our I/O resources. I also decided that a one 100 Mb SAS data set would be a good size file for the test. The Proc Sort had 4 by variables. Memory and cache was cleared between each run. As it turned out, the test selected led to an important discovery about how Proc Sort works and about how to speed it up. More on that a little later.

Different combinations of the above equipment were assembled and over 700 tests were run. Proc Sort was tested on single drives and RAID 0 arrays of 2 and 4 drives. It was mentioned earlier that RAID levels 1, 0/1 and 5 were also tested but only briefly. Software RAID levels as well as Hardware RAID levels was tested. The stripe size was altered on all combinations in order to see if it would have any effect on performance. This was done since there is speculation that this can affect performance. Stripe size is the size of the block of data that will be written in parallel to each drive in the RAID

array. Stripe size is most often referred to as blocksize. See appendix one for more information on Stripe size.

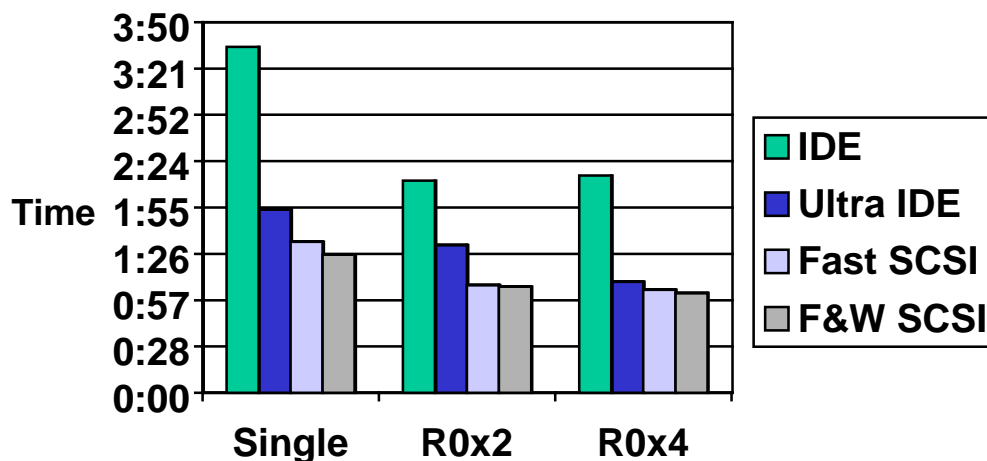
What the Testing showed

In the following graph, you will be able to see that the slowest hard drives were the Standard IDE hard drives. The slowest time was not put on the graph since it was turned in on the slower of the two computers used in the test. As a matter of fact, the only scores used were the ones turned in on the faster computer so that a good comparison could be made between all combinations tested.

The next faster hard drives were the Ultra IDE hard drives. However, there was a remarkable improvement of 45% over the Standard IDE hard drives when these drives were tested on an Ultra IDE interface card. When the Ultra IDE hard drives were attached directly to the motherboard's IDE slots, there was still a 36% gain in performance over the Standard IDE hard drives. What this indicates is, for a lot of older servers whose motherboard doesn't support the new Ultra IDE standard, you may be able to add Ultra IDE hard drives and still get some pretty good performance without having to add a new IDE interface card.

A note about the Ultra IDE hard drives that were used in testing. I didn't have the fastest Ultra IDE hard drives that were on the market. The IBM Deskstar had that reputation when I was doing the testing. Since then, Seagate has introduced a new line of Ultra IDE/ATA hard drives that are setting new performance records. Their advantage comes from their 7200-rpm spindle speed. This is the same spindle speed of the Fast SCSI and Fast and Wide SCSI hard drives that were tested. Remember the faster the data moves under the head, the less latency there will be and the transfer rate will be higher. In the Sort Times graph, Single represents the data, SASWORK subdirectory and the

Sort Times



paging file being on one single hard drive. R0x2 is a RAID 0 array with two hard drives in the array while R0x4 is a RAID 0 array with four hard drives in the array.

The SCSI hard drives gave the best overall performance. Testing Statistics and comments will be presented next. But first, one thing that needs to be pointed out is I found that by moving the SASWORK subdirectory to a different hard drive, I could affect the sort times tremendously. Keep this in mind as you read through the next section. I found that having the paging file on a different hard drive didn't make that much difference. However, I would speculate that a server that had lots of things going on during the sort, probably would benefit from having the paging file on a hard drive of it's own. I also found that duplexing helped to achieve the fastest sorting times but was not a big factor in our performance testing. Also you will notice that hardware vs. software RAID gave mixed results.

Testing Summary and Comments

The **fastest time** was 00:57.8. It was achieved with two NT software RAID 0 arrays. Each array was a striped set of two Fast and Wide Seagate (ST34572W) 4.5 gig SCSI hard drives. SASWORK was on one striped set and the data and the page file was on the other striped set. Stripe size was 64k. Drives were formatted with NTFS. Two Adaptec 2940UW's were used. Two drives per controller.

The **second fastest time** was 00:58.2. It was achieved with an Adaptec RAID controller (AAA-131 with 1-MB cache on it). A hardware RAID 0 array of four Fast and Wide Seagate (ST34572W) 4.5-GB SCSI hard drives and an NT software RAID array of four Fast Seagate (3/ST32272N's & 1/ST32171N) 2.2 gig SCSI hard drives. The Fast SCSI's were on an Adaptec 2940UW controller. SASWORK was on the Fast SCSI array while the page file and data were on the hardware RAID array.

The **third and fourth fastest time** was 00:58.6. This was shared by two different setups. **First one** was an Adaptec RAID controller and two hardware RAID 0 arrays with two Fast and Wide Seagate (ST34572W) 4.5 gig hard drives in each array. SASWORK was on one array while the page file and data were on the other.

Second one was the same setup as the fastest time. The data and SASWORK were switched so that the SASWORK was on the same array as the page file. I really don't think a case can be made that the SASWORK shouldn't be on the same drive as the paging file in order to achieve the fastest times. My reasoning is, there were no other demands on the server, and therefore it didn't have to page or use the paging file. That's why throughout the testing we didn't see any improvements in speed when the paging file was on a drive all by itself. But the situation would change drastically if you have other demands on the server for sure. If the OS had to page, then having the paging file on a different drive would certainly help cut the sorting time.

Fifth, sixth and seventh fastest time was 00:58.7. Shared by three different setups.

First one was same configuration as fastest time. Page file and data on one array and SASWORK on the other.

Second one same as the second fastest time. Just switched the data and SASWORK.

Third one was almost the same setup as the second fastest time except I moved the paging file to the slow standard EIDE hard drive that the NT system was on.

Eighth fastest time was 00:58.8. It was achieved with an Adaptec RAID controller (AAA-131 with 1-MB cache on it) and two hardware RAID 0 arrays. Each array had two Fast and Wide Seagate (ST34572W) 4.5 gig hard drives. The paging file was on one array and the SASWORK was on the other. A third NT software RAID 0 array of four Fast SCSI Seagate (3/ST32272N's & 1/ST32171N) 2.2 gig hard drives were used to pull the data from.

Ninth fastest time was 00:59.0. This was achieved with the same setup as the eighth except the data was moved to one of the hardware RAID 0 arrays which had two Fast and Wide hard drives. The SASWORK was moved to the NT software RAID 0 array, which had four Fast SCSI drives.

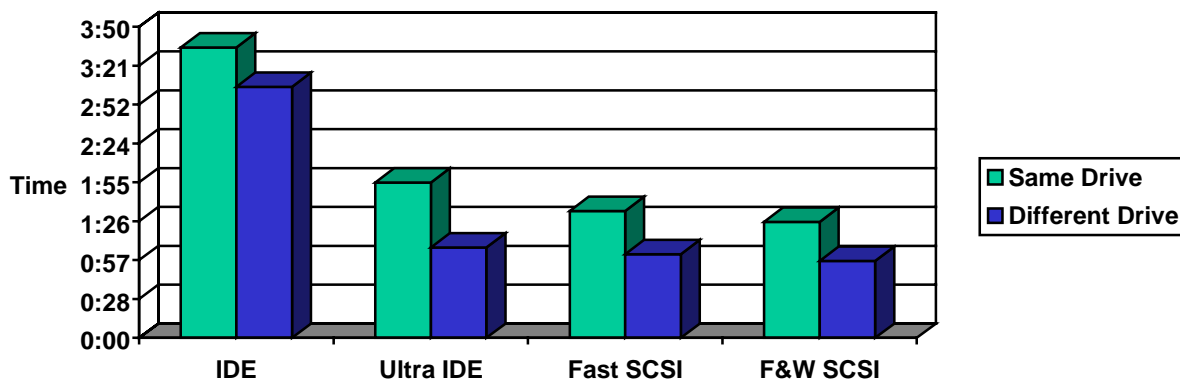
Tenth fastest time was 00:59.1. Same setup as eighth fastest time. This was just another run. I found the times would vary ever so slightly like this when they were repeated with the same setup.

The **slowest time** was 03:35.0. This can be seen by looking at the Sort Times chart above. It was turned in on a Standard IDE hard drive that was attached directly to an IDE port on the motherboard. This was 59.8% slower than the fastest single drive speed turned in by a single Fast and Wide SCSI hard drive. The Ultra IDE hard drive was only 24.1% slower than the Fast and Wide SCSI hard drive while the Fast SCSI hard drives were only 8.4% slower.

When the hard drive combinations were configured as software RAID level 0 with 2 hard drives in each array, the standard IDE hard drives appeared to benefit the most with a 36% increase in performance. Fast SCSI was next at 28.7%, Fast and Wide SCSI followed with 23% and the Ultra IDE hard drives brought up the rear with a 19.2% increase. All in all, these increases in performance represent solid performance gains. What was surprising though was when two more hard drives were added to each array to make a RAID level 0 with four hard drives in each array. The Standard IDE hard drives actually lost ground (one of the hard drives died and was replaced with a slightly lower performing Standard IDE hard drive which might explain this) while the Ultra IDE hard drive array gained the most ground with a 25% increase in performance. The Fast SCSI and Fast and Wide SCSI hard drives gained only 2% each in performance.

The BIGGEST difference

Moving the SASWORK subdirectory to a hard drive by itself made the Biggest Difference in performance as the chart below will show. Tests were done on each hard drive type with the paging file, data and SASWORK subdirectory residing on one drive. Then the SASWORK subdirectory was moved to another identical hard drive.



The best results were turned in by the Ultra IDE hard drives with a 41.4% increase in performance. Fast SCSI was next with a 34.1% increase. Fast and wide was very close to the Fast SCSI drives with 31.1% increase. The Standard IDE drives came in last with an 11.1 increase in performance. Being able to read the data from one drive and write the utility file that Proc Sort uses to a different drive gave us huge performance gains. I also tried this with RAID 0 arrays of two and four drives. I found that I could increase the performance but couldn't get the large increases that we saw with single drives. I concluded that this was one super way to get good solid performance gains. And it doesn't require the most expensive equipment to get that performance either.

The Big Discovery

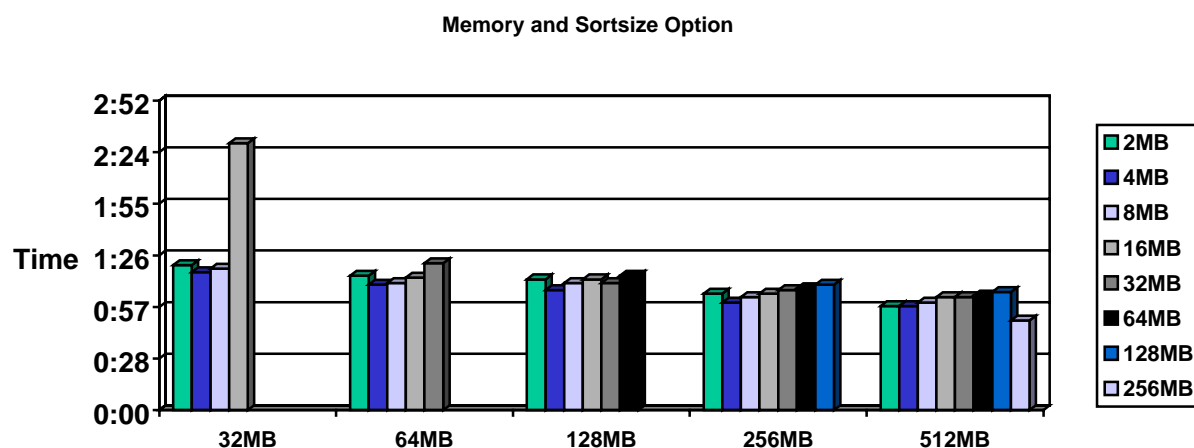
Affects of memory on sorting

I also looked at how memory can affect sorting. Anyone that has worked with the SAS system for a while knows that there is a **sortsize** option that can be used to limit the amount of memory that will be used to sort data in. The default sortsize is 2 megabytes. To verify this, edit the Config.SAS file and you should see '-sortsize 2m' on a line by

itself somewhere before the ‘Do not edit below this line’. You can also issue Proc Options short; run; from within the Display Manager and look in the log file to see what it is currently set to. Sortsize can be changed in the Config.SAS file or it can be passed in via an options statement while running a job from within Display Manager or non-interactively as a batch job. Ex. Options sortsize=4m;

Arguments about whether or not this option helps have gone on for years. So, it was important to test this option to see exactly what effect it would have on sorting the 100MB data set.

Memory is expressed in the graph below along the bottom row. That is, the memory that the PC was allowed to use. The data was then sorted with different sortsize options per memory size to see how it would affect the time it took to sort the data set. The vertical



legend on the right hand side shows the color of the different sortsizes used. No more than half of the available memory was ever used for the sortsize.

As mentioned earlier, sortsize defaults to 2MB. Looking at the graph above reveals that a sortsize of 4 and 8 MB yielded a slightly faster time in most cases until a sortsize of 256MB and 512MB of memory was allowed to be used on the machine. Then at that point I noticed that the highest value (which usually yielded the slowest sort time) was all of a sudden faster than all the other sortsizes. This led to an important discovery about the sortsize option. The best sort time previous to this last test had been 57.8 seconds. The best time now was 39.7 seconds. So, the question became, ‘Why did having a huge amount of memory and using a huge sortsize allow the sort time to be 31% faster than our previous best sort time?’.

It was time to call the developer and ask some questions. And this is what I learned. The way the sortsize option works is to limit the amount of memory that Proc Sort can use. If the data set is larger than the amount of memory that Proc Sort is allowed to use, then Proc Sort will create a temporary utility file in the SASWORK subdirectory. Proc Sort then request data up to the memory size specified by the sortsize option. The data is sorted in memory then is written out to the utility file in the SASWORK subdirectory.

The process is repeated until the whole SAS data set has been sorted. Proc Sort then merges the results that are in the utility file into the final data set. If there are no errors then there are several pathways as to what happens and where the final sorted data set is placed. If there was no out= option on the Proc Sort statement, then the final sorted data set is placed in the same directory as the original data set that was being sorted. The original data set is deleted and the final sorted data set is renamed the same as the original. If there was an out= option on the Proc Sort statement, then the final sorted data set is placed where the out= option directs it. It is also named according to what is in the out= option.

Now the key here is the sortsize option **limits** the amount of memory that Proc Sort can use. Since most computers have limited amounts of physical ram, this is a good thing. However, what if you had more than enough memory to sort the data set in? You could assign a sortsize large enough to hold the whole SAS data set in memory. Then a temporary utility file in the SASWORK subdirectory wouldn't be necessary. That would eliminate the I/O associated with writing to the utility file, which would save time. Instead of having to write to the utility file, Proc Sort can now just write directly to the output data set from memory when the sorting is finished. And that is exactly what happened in the test. But that wasn't the entire story.

I tried limiting the memory yet keeping the sortsize and memory large enough to hold the whole sorted data set just to see what would happen. To do this I had to first learn to figure the amount of memory that would be needed to hold the 100MB data set in memory. With a little help from my developer friend I came up with the following formula:

1. Use Proc Contents to get statistics on the SAS data set to be sorted.
2. Write down the length of each by variable that Proc Sort will use. If the length is not on an even 8-byte boundary, then round up to the next even 8-byte boundary.
Example: A 9-byte variable would be rounded up to 16 bytes.
3. Add the length of all by variables (remember to use the rounded up figure/s).
4. Add this figure to the length of one observation (which you can get from the Proc Contents report).
5. Multiply the result by 1.1.
6. Multiply the result by the number of observations (which is also in the Proc Contents report).

This will give you the approximate amount of memory (plus a little extra) that will be needed to hold the sorted data set in memory and eliminate the need for a utility file.

Based on this formula, the 100MB-test data set required 171MB of memory. So, I limited the machines memory use to 256MB of physical ram and reran the Proc Sort again. The sorting time increased to just over 50 seconds even though the whole data set was now in memory and no utility file was being created. This was faster than the

previous fastest time of 57.8 seconds but slower than the current fastest time of 39.7 seconds. Why?

To answer this question I started watching Task Manager to see just what was happening to the memory. At first this didn't tell me anything. So, I upped the memory to 384MB and like magic the sort time was reduced back down to 39.7 seconds. What I observed with Task Manager was, the system started out with 354MB free (after NT, all services, the SAS System and everything was loaded). Once the sort started, memory dropped to 202MB and then started rapidly declining until there was only 2MB left. Then very quickly the memory started climbing until it was back to the 354MB that it started with. Why was 180MB of memory above what I figured it would take to sort the data set in used? I theorized that Windows NT must have been sensing that the application was reading in a sequential file and would probably need all of it eventually. So, it must have started reading ahead and caching the file so that when the application needed more data, it could be read from memory. I later confirmed in the Microsoft Knowledge Base that that was exactly what NT will do. If it is able to develop a pattern for what is being read, then it will start read ahead caching.

But wait. That should have only used up 100MB because that's how large the SAS data set was. What about the other 100MB of memory that was used, where did it go? Again the Microsoft Knowledge base provided the answer. Windows NT can do what's called Lazy writes. Which means that it will cache writes until the system is less busy and then write them out. Well, the writes were coming too fast so the data just kept piling up in the cache while NT waited for the system to be less busy and get a chance to write. Since there was plenty of memory to write to, and since the system wasn't busy with other applications, it waited until Proc Sort was finished. At that point, the whole data set was written out to disk in one fell swoop.

Closer investigation revealed that Proc Sort only needed 152MB to sort the data in and Windows NT used just over 200MB to cache the reads and writes. That left the system with about 2MB of memory at the lowest point. But this didn't matter since there were no other applications running. I concluded that my formula for figuring the amount of memory that Proc Sort would need to sort the data set was very conservative. In this case it was about 20MB over. But that didn't hurt since Proc Sort requested only what it needed. I also concluded that you would need memory about 3.5 times the size of the data set to get the performance to this level.

I asked again about the extra amount of memory that the formula allowed for. I was told that depending on the data, number and type of variables, it would be good to have this extra overhead. Especially since the memory wouldn't be used if it weren't needed. But using the formula just the way it is would always insure that you allowed enough room in your calculations and wouldn't come up short during the sort. We have already seen the graph showing the sorting times. If you were to come up short on memory, you would actually run slower than using lesser amounts of memory. So, always figure the extra margin in by using the formula as is. If you have other applications running on the machine, you would also have to take into consideration how much memory they were

using in order to figure out if you had enough memory to get the maximum performance for sorting.

So, sorting a 100MB data set has been explored, but I don't dare come to the conclusion that this is the final word on sorting. However, now there is a way to determine the amount of memory you'll need to avoid the utility file in the SASWORK subdirectory. This alone should save time by negating the I/O necessary to write to the utility file. Some insight into the way that Windows NT uses caching (read aheads and lazy writes) to increase performance has also been presented. Armed with this information, you should be able to do a little testing and hopefully find ways to get more performance when sorting your SAS data sets.

Conclusion

Buying a new computer can be a challenging task for anyone. Especially, if it is to be placed into use as a server for your company. As profit margins have fallen for electronic components, marketing hype has exploded making the task even more formidable than before. Whose ads do you believe and how do you size the equipment for your workload? Who is going to set the equipment up and even more importantly, who is going to tune it so that your new computer processes it's workload as quickly as possible?

If you have read this paper carefully, you are now a more informed buyer. You also understand that any new computer that you buy and place into service will have to be properly tuned in order to achieve it's potential. You also understand that your existing equipment has the possibility of being tuned to achieve a higher level of performance. But understanding doesn't stop here. It is also vitally important to understand that as your company grows, so does it's computing needs. This will add increasing demands on your computer system and performance will become a larger and larger issue until it becomes evident that a better computer system is what's really needed.

There are many avenues that can be explored in the quest to increase the performance of existing equipment. Some of them may involve simple reconfiguration steps. Others may involve running utilities that can restore then maintain the level of performance that you started out with. Still other ways may involve new expenditures on higher performance equipment. But how do you know what to address? I have tried to provide some information that can be used to find out where the bottleneck in your system is. Performance monitor is an excellent tool and should be studied in more detail so that you can utilize it to it's fullest potential. But don't stop here. Look for other tools that you can use to analyze your computer system and get the information you need to make the proper tuning adjustments.

Drawing on my eleven years of experience working with SAS System users, and the testing that I did for this paper, I have stated that a lot of our customers are faced with I/O constraints. I confined my testing to the small server arena since larger shops are more

likely to have Windows NT Administrators on their staffs that have been trained to tune and administer to the needs of Windows NT servers. I have shown that there are inexpensive ways in which to improve the performance of the SAS System such as moving the SASWORK subdirectory to a hard drive of it's own. I have stated that moving the Windows NT paging file to a hard drive of it's own may also help. My research has shown that the new advances in the Ultra IDE/ATA hard drives are certainly something that should be looked into. The overall I/O performance on your system can be boosted by utilizing them to put the SASWORK subdirectory on and/or the Windows NT paging file. And, this can be one of the least expensive ways to boost performance. This is due of their excellent cost to performance ratio over the current performance leading SCSI hard drives. As the newest Ultra IDE/ATA generation of hard drives enters the market, they will make even greater impact as their cost to performance ratio continues to improve and race ahead of SCSI hard drives.

During my research, new ways to use large amounts of memory when sorting SAS data sets were discovered and has now been delivered to you through this paper. Implementation of this information is up to you. Memory is at an all time low right now cost wise and continues to fall every day. This will make it possible for the smaller shops to afford more physical ram with which to increase performance on their servers. I believe that this will help you achieve moderate to dramatic increases in performance if properly implemented and utilized.

A lot of other tid-bits about performance have been presented. Please don't dismiss them for there may be that hidden gold nugget among them that you've been looking for.

Finally, it is my sincere wish that within the body of this paper you would find something that will help you increase the performance of your computer system. If that is not the case, then I hope that the information contained here will help you to understand why and give you some ideas as to what you can do about it.

Appendix A

Performance Tips

1. Defragment your hard drives on a regular basis. Executive Software offers a product called Diskeeper. Symantec also makes a disk defragmenter available in Norton's Speed Disk.
2. Use Error Checking to scan the hard disk for errors on a regular basis. You can do this from Disk Administrator. Just click on Tools/Properties/Tools. Select Error 'Error Checking'. In the Error Checking dialog select both options. If you receive a message that tells you that disk checking can't be done right now because the disk can't be locked for exclusive use, select 'Yes'. Reboot your machine and the disk will be checked before NT comes up.
3. Put the Windows NT swap file (virtual memory) on its own hard drive. You don't have to go out and buy the most expensive hard drive either. If you've read through this paper, you should now have a better appreciation for the new Ultra IDE/ATA hard drives that are out on the market. The newest 7200 RPM drives are going to stir things up even more since their performance to price ratio is outstanding. Format the swap drive as FAT and select an allocation unit size of 64k. To do this you will have to use the format command and the /A: switch at a CMD prompt.

Example: `format d: /A:64k /FS:Fat`

The `d:` in this example stands for the D drive but could be any drive that you have set up to be the new swap file drive. Just use the appropriate drive letter instead of `d:`.

The `/FS:` switch tells the format command to format the drive as FAT. Some of you may be wondering why the drive should be set up as FAT. This is due to the fact that Fat drives are usually faster than drives formatted as NTFS. Since you won't have to be worried about security on this drive, why not go for the performance of FAT?

Specifying an allocation unit size of 64k will also help performance slightly. When using the `/A:` switch with the format command, you cannot specify more than 64k as the allocation unit size if you format the drive as NTFS. But by formatting the drive as FAT, you can select allocation unit sizes of 128k or 256k. A larger allocation unit size usually means better performance. This is not always the case however. When I checked the throughput on a FAT formatted hard drive, the best blocksize was 64k.

Another note here about allocation unit sizes when formatting hard drives and stripe sizes when formatting RAID 0 drives. I used all the different settings and found that 64k and 128k were the two best settings to use. See the Microsoft NT Resource kit for more information on allocation unit sizes and stripe sizes.

4. If you don't want to purchase another drive just for the swap file, you might consider allocating a separate partition to set the swap file up in. This partition could also be formatted as FAT with a large allocation unit size. Since the swap file will be on the same drive that other partitions are on, you may not gain the speed you would by having a separate drive for the swap file. However, by using a FAT partition with a large allocation unit size, you will hopefully see some performance gain. And since the swap file will be the only thing using this partition, you won't have to worry about fragmentation from other applications.
5. Set the SASWORK subdirectory up on it's own hard drive and format it as FAT with a large allocation unit size. This should enhance the performance of the SAS System even more than putting the swap file on a separate hard drive. That is, if your server is not encountering a lot of other traffic or paging a lot. If it is, then putting the swap file on it's own hard drives is definitely a good idea. Testing here at SAS Institute has shown over and over that having the SASWORK subdirectory on a drive that is different than the drive that the data is on enhances performance.
6. Set the swap file up as a permanent file. This can be done from the Performance tab inside System Properties. Just make the Initial size the same as the Maximum size. This will keep the swap file from becoming fragmented by other system writes. If you set up a separate partition or hard drive for the swap file, this is not necessary. However, it still might be a good idea since the system won't have to use up CPU cycles to dynamically resize the swap file.
7. Use Mirroring as opposed to RAID 5. If you use RAID 0/1, you can also get very good performance. This will provide faster writes as well as reads as opposed to RAID 5 which has to compute and write the parity information to several hard drives every time something is written to it. There are two problems with mirroring. One is cost. This solution cost twice as much and the second problem is the hard drives must be exactly the same. If you run into a hard drive that has some bad sectors, then the mirrored pair is different in size. Since one drive is smaller, true mirroring cannot take place and will probably fail.
8. Use Performance Monitor to determine if adding more memory would be wise.
9. Remove all wallpaper and Screen Savers as they can slow performance. The Screen Savers with 3d graphics are the worst ones. They will use up quite a few CPU cycles while building the graphics that show on the screen. If you're worried about your monitor and screen burn-in, you can turn it off or use the Energy Saving Features in the Display Properties Screen Saver tab.
10. Remove all Services that are being started but not utilized. They just sit there and waste the memory that they're stored in.
11. Buy faster hard drives or better yet a bundled RAID system.

12. Upgrade to faster processors if possible. Or, add more processors. Be sure to use Performance Monitor first to see if adding more processors would help.
13. Turn on write buffering on SCSI drives. This might help especially on RAID 5 Systems. SCSI drives are usually shipped with write buffering turned off. This keeps from losing data if the buffered writes have not been written to the hard drive yet and the system is suddenly turned off.

Warning: SAS Institute, Inc. will not assume responsibility if you lose data by using write buffering. Please be advised that you may lose data in certain situations if you try this.

14. Install an electronic hard drive. These drives are nothing more than a large amount of memory that has been put together and made to work as a hard drive. Very similar to a ramdisk. Since they are extremely fast to access, virtual memory and/or the SASWORK placed on these drives would speed the system up tremendously. But it doesn't make sense to buy one of these drives in lieu of buying enough ram. Having enough physical ram can keep you from excessive paging. However, there is always going to be some amount of paging regardless of the amount of physical ram you have. So, if you have plenty of physical ram and want to get even more speed, consider trying one of these drives for the virtual memory and/or to put the SASWORK subdirectory on.
15. Load only the network protocols you need as any extra ones will only take up memory and slow performance.
16. If any of your users are running 16-bit legacy applications on your application server, have them set up a Shortcut on their desktop from which to run these applications. Right click on the Shortcut icon and select the Shortcut tab. Click the box that says "Run in separate Memory space". What this will do is create a VDM that will be specific for that application. This gives the application it's own memory space to play in and improves performance and stability.
17. If you have CPU intensive applications and can schedule them to run on off-peak hours then by all means do so. These applications compete for resources, which will slow everything else down and they can easily be scheduled using the AT scheduler that ships with Windows NT. A GUI based AT scheduler is included in the Windows NT Resource Kit.
18. If you are using older 8-bit network adapter cards, consider upgrading to 16 bit or even 32-bit adapter cards as these cards do not use as much processor time.
19. If you have determined that the CPU is a bottleneck to the type of applications that your server is doing, upgrade to a faster CPU if possible before considering adding more CPU's. Remember that there are not a lot of SMP (Simultaneous Multi-

Processing) applications out there that can make the most out of dual (or quad) processing systems. **The SAS System is not SMP compliant.** That is, the SAS System **cannot** take advantage of multi-processor systems.

20. Monitor your applications to see if any of them leak memory. Contact the manufacturer of the application for updates or corrections if you find any leaks.
21. Remove unnecessary Network Protocols since they will take up space in the paged and non-paged memory pools.
22. Keep a list of drivers installed on your server and the companies that wrote them. Monitor those companies on a regular basis to see if newer releases of the drivers become available. My experience has shown that installing newer drivers can boost performance in many cases.
23. Check the following location in the System registry hive to ensure that your current L2 cache is set correctly. The usual amount of L2 cache is 512k bytes. However, your server may have a different amount of L2 cache. If the current setting for L2 cache is smaller than the amount of L2 cache your system has, performance can suffer.

HKEY_LOCAL_MACHINE\System\Current\ControlSet\Control\SessionManager\MemoryManagement\SecondLevelDataCache

Warning: SAS Institute, Inc. will not assume responsibility if you decide to edit the Registry. If you decide to try this tip, please make a back up before you proceed.

The hexadecimal value for 512 is 200. Two Hundred is what shows up in my registry when I check this entry. I can toggle it to decimal and it will then report 512.

24. Another registry entry you can try that may improve performance on your server is:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SessionManager\MemoryManagement\DisablePagingExecutive

Warning: SAS Institute, Inc. will not assume responsibility if you decide to edit the Registry. If you decide to try this tip, please make a back up before you proceed.

By default it is set to 0 which allows NT to page drivers and the kernel. If you set this value to 1, then the kernel and drivers will be kept in active memory and never paged out. One precaution though. Use this technique only if you have plenty of memory. That is if you have 512 MB of memory or more. It may be effective on systems with smaller amounts of memory. Be sure to test it well if you do use it on systems with smaller amount of memory though.

25. There are performance problems associated with the Findfast utility that comes with Office 97. If you have installed Office 97 on your server, then the Findfast utility will run every two hours by default. You might have experienced this as a flurry of hard drive activity every two hours. While the hard drive is being continuously used, the CPU/s is/are being utilized by Findfast so that it/they can reindex all Office documents. Findfast also runs every time someone uses File/Open from within a Microsoft Office application. I think it would be best to stop Findfast totally. But there may be situations in which it would be best to keep it running. You'll have to be the judge and jury on this. You can find more information in the following Microsoft Articles:

PSS ID Number: Q158705 and PSS ID Number: Q170610

The default two-hour setting can be changed so that it reflects the needs of your users yet doesn't strain your resources. However, you will have to experiment with different settings until you find the right balance. To change this setting, open the Control Panel and double-click the Find Fast icon. Now click Index then click Update Interval. In the dialog that appears you will be able to confirm that the Update Interval is set to 2 by default. Type in a new interval and click OK.

26. Turn off the Autorun feature so that your system isn't constantly polling the CD-Rom drive to see if a CD has been inserted. This event just wastes the small number of CPU cycles it consumes. To do this, you will have to edit the Registry. The entry that you will have to change is:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\CDRom

By default, this entry is set to 1, which enables the Autorun features. Set it to 0 in order to disable it.

Warning: SAS Institute, Inc. will not assume responsibility if you decide to edit the Registry. If you decide to try this tip, please make a back up before you proceed.

27. SCSI hard drives, and to a lesser degree some of the new Ultra EIDE hard drives, produce a lot of heat. This excess heat can reduce total hard drive performance by as much as 25% or more. A way to counteract this is to employ more exhaust fans that pull air through the case and to put extra fans inside that can blow directly on the hard drives. You can also put hard drive coolers on each hard drive when possible.

Appendix B

Tuning Memory on a Windows NT 4.0 Server

There are two main ways of tuning memory on a Windows NT 4.0 Server. First of all, you can specifically tell NT server how you want it to use memory. That is, do you want it to be responsive to fewer than 10 users? Or do you want it to be responsive for up to 64 users? Do you want it to be more responsive to file sharing? Or do you want it to act as an applications server? The role you want your server to play depends on your needs of course. But NT can't play the role you want it to play without input from you. The choices are:

1. Minimize Memory Used

If you select this one, your NT Server will use conserve memory used for handling network-based request. If you expect less than 10 users and this server is to be your Primary or Backup Domain controller, then this is a good choice. It's also a good choice for running a lot of foreground applications if you're running NT Server instead of NT Workstation on a single user machine.

2. Balance

Default setting for Primary or Backup Domain Controller. Don't change this setting unless you think that there will never be more than 10 users on the server simultaneously. If this is a member server, then this setting would allocate memory for up to 64 users.

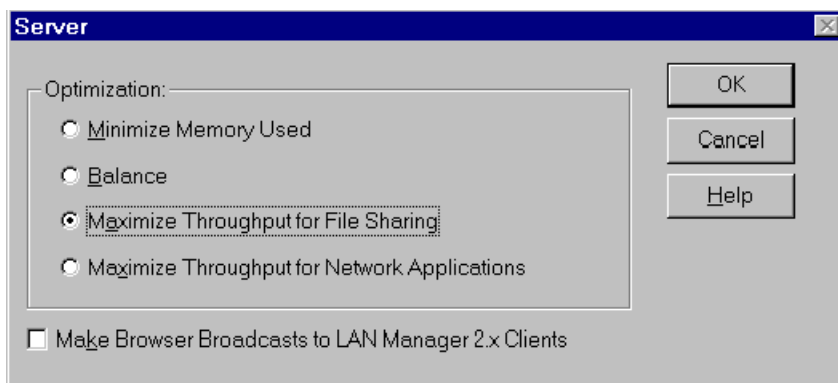
3. Maximize Throughput for File Sharing

Default setting for a member or stand-alone server. Concentration is on optimizing memory so files can be delivered over the network quickly. Memory is allocated for large disk cache areas. Maximum memory is reserved for use in file sharing operations. This setting gives the dynamic cache higher priority for memory than the working sets of applications.

4. Maximize Throughput for Network Applications

Memory is optimized for distributed applications, especially those that do there own caching in memory. This setting gives the working sets of applications higher priority for memory than the dynamic cache. An example of a good choice for this setting would be a database server.

In order to configure your server's memory properly, click on Start, and then select Control Panel. Double click on "Network" in the Control Panel dialog, then select "Services" from the Network dialog. Finally, double click Services and the Server dialog will appear. Here is an example of what the Server dialog looks like.

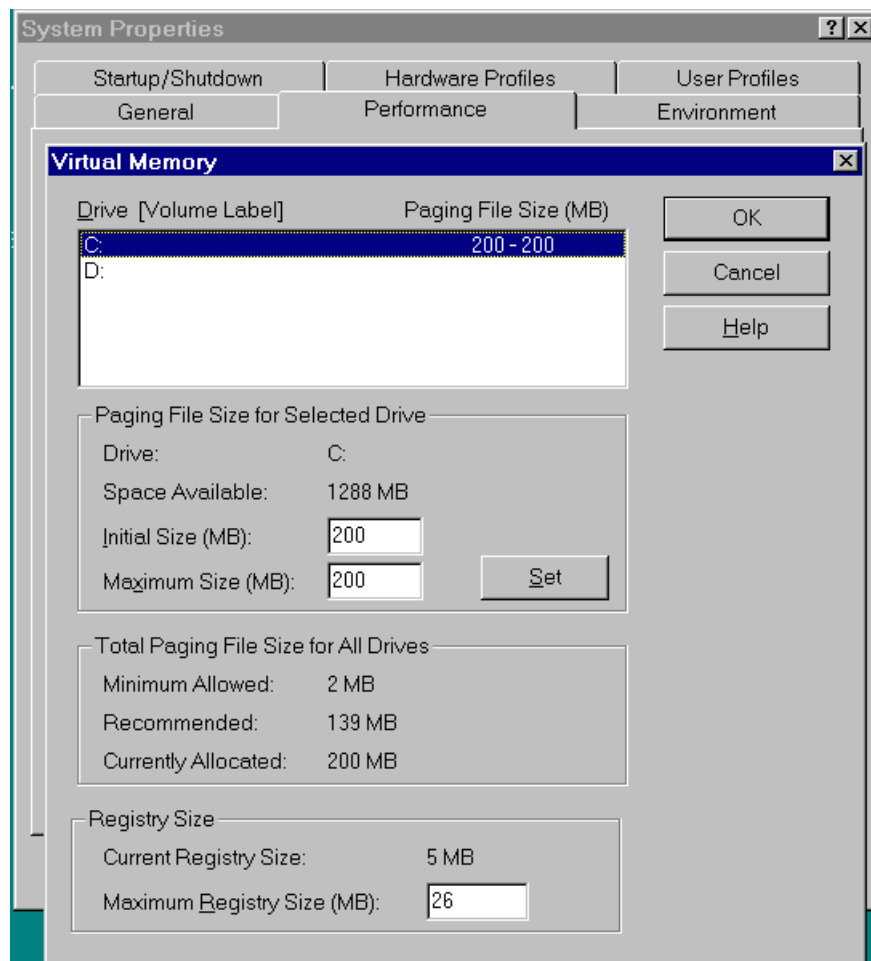


Note: You won't find this dialog available on Windows NT Workstation. It is only available on Windows NT Server.

No one can tell you what the best setting for your server is. You will have to make this call yourself. If you're not sure, then you may want to do some experimenting with the different settings in order to determine which one is right for you.

The second thing that you can do to tune memory is to configure the paging file optimally. Notice I didn't say virtual memory? One of the things I need to address right off is, virtual memory does not mean paging file or swap file although it does include the paging file. Time and time again I have talked to people who get this confused. I think that it comes from the old Windows 3.1 days when virtual memory usually meant the paging or swap file. Also, if you go into Windows 95 to tune the memory, the dialogs talk about the hard drive space as virtual memory. That may be on Windows 3.1, 3.11 or Windows 95 but not to Windows NT. Virtual memory is the physical ram as well as the paging file to NT. In other words, the total address space that Windows NT can use. That's why the paging file is named pagefile.sys and not virtualmemory.sys. Just for the record, Windows NT uses 32 bit addressing which allows a total of 4 gigabytes (2 to the 32nd) of memory to be addressed. Two gigabytes are addressable by applications and the other two gigabytes are reserved for the operating system.

In order to tune the paging file, move your mouse pointer to the 'My Computer' icon at the top left of your screen and right click on it. A dialog will pop up from which you will select Properties. Now select 'Performance' from the System Properties dialog that just came up. In this dialog, you will find a block called Virtual Memory. I note that even Microsoft doesn't make it easy to understand the differences between Virtual memory and paging file. However, in this Virtual Memory block you will find 'Total paging file for all disk volumes' and a 'Change' button. If you click the 'Change' button a 'Virtual Memory' dialog will appear in which you can change the size and location of the paging file. The paging file can be located only on one drive volume or it can be located on several drives. How many places it can be placed on depends on what's listed in the 'Drives [Volume Label]' section. As you can see in the example below, there are two possibilities.



The paging file could be set up on C: or D: but is setup as a 200MB permanent file on the C: drive above.

Possibilities:

It could be setup only on the D: drive.

It could be set up on both drives C: and D:

This would provide space on D: for Windows NT to spill the paging file onto when space on C: ran out. I have also talked to many people who feel that this has provided them a performance boost. The reasoning is simple enough. NT can be reading and writing to both drives at the same time.

To increase the existing paging file on C: you could increase the Maximum Size (MB): in the 'Paging File Size for Selected Drive'. Or you could click on D: and set the Initial Size (MB): and Maximum Size (MB): to add a paging file on D: that Windows NT could use when it ran out of room on the C: drive. Finally, you could setup a paging file D: then click on C: and set the Initial and Maximum sizes to 0. This would delete the paging file on C: so that the paging file would then reside only on D:. You would have to reboot

to complete the process. Notice that the paging file above has been set to an Initial Size of 200MB and a Maximum Size of 200MB. What this does is tell NT to make the paging file static. Meaning, NT will not dynamically resize the paging file as needed to accommodate the applications it's running. The advantages of doing this are the paging file will not become fragmented since space is now reserved only for the paging file. This helps to fight performance degradation especially when the computer is not shut down very often.

Appendix C

Feedback on a hardware suggestion

I talked with one of our users at SUGI this year that was having problems with performance on one of his client's servers. It was a small server with a RAID controller and 3 Fast and Wide SCSI hard drives setup as RAID 5. SAS applications were taking between 2 to 2.5 hours to complete. I suggested that a IDE RAID controller configured as RAID 0 with two EIDE 6.4 gig hard drives be set up. Then move SASWORK over to it. He followed my suggestion and their SAS application times came down to 20 to 40 minutes. It was a dramatic increase in performance. But there were problems with data corruption. After many conferences with manufacturer of the IDE RAID controller and BIOS upgrades, a technician at his shop shortened the IDE cable to 10 inches. Viola, the data corruption problems went away. It was a solution that no one had anticipated, not even the manufacturer of the IDE RAID controller. So, the question became, 'Why did shortening the cable help?'

I can only speculate at this point but I'd be willing to bet on this. The copper in the IDE cable was not pure enough to support such a high data transfer rate. So, shortening the cable cut down on the amount of impurities in the copper that the data were exposed to.

My speculation is based on network cabling. When you connect a 10 megabit/second network card to the network hub, you don't need a high grade network cable (providing you stay within the distance the cable is rated for). But if you connect a 100 megabit/second network card to a network hub, you must use a grade 5 (high quality) cable. These cables have a copper content that is supposed to be a lot more pure than the standard lower grade cables. Without these high grade cables, you can't get the network connection to work properly.

A standard IDE cable can have a length up to 18 inches for hard drives. The documentation for every hard drive manufacturer that I have ever read warns about using

cables longer than this. I think that they probably know that data corruption will occur if the cables are longer. I have found IDE cables that were 18 inches long. And I have found cables that were up to 40 inches long. But I have never found IDE cables that were advertised as high-grade cables. Maybe they don't exist. Maybe the new transfer rates of Ultra EIDE are so new and unexpected that the industry hasn't prepared properly. Whatever the reason, it's time for the computer industry to address this issue since 10 inches doesn't allow for easy installations in most server cases. If high-grade IDE cables shipped with the IDE RAID controllers, consumers would never encounter this problem.

So, why didn't the manufacturer run into this problem during their testing of the product? Again I can only speculate. Maybe the manufacturer had better cables. Or maybe they did most of their testing with four drives instead of two. With four drives, RAID 0 would stripe the data across all drives. Since the drives are connected by two IDE channels and two cables, 50% less data would travel over each channel thereby reducing the speed that the each cable had to deal with. Maybe the user had cables that were not as good as standard cables. Or maybe there were other extenuating circumstances. We'll never know. But the user assured me that the manufacturer in this case did all they could to help. The user said that they just happened on the solution before the manufacturer.

Appendix D

Alternate Resources

No paper would be complete these days without pointing the reader to some resources that can provide more information about the subject. More and more these days these resources are on the Internet. While I think the Internet is great, I have one problem with it. Any links I put in here will either move or disappear over a given period of time. Still, it's better to have them than not have them. So, here you are. They're in no particular order as they're all very informative links.

www.microsoft.com	www.venus.it/homes/spumador/infohw.htm
www.anandtech.com	www.cern.ch/HSI/fcs/storage.htm
www.pricewatch.com	www.zdnet.com/zdhelp/static/besttips_net.html
www.sysinternals.com	www.plazaone.com/allwindows/winnt.htm
www.pctoday.com	directory.compuserve.com/Forums/forumsP.asp
www.cmpnet.com/	www.digiweb.fr/~chadwick/
cma.zdnet.com/	www.lemig.umontreal.ca/bios/bios_sg.htm
www.execsoft.com/	thef-nym.sci.kun.nl/cgi-pieterh/atazip/atafq.html
www.bhs.com/	www.flash.net/~hollowel/pro/index.html
www.techweb.com/	www.storagereview.com/
www.pcguide.com/	www.winntmag.com/Redir/Redir.cfm?Area=/Update

