



Gregg McKnight
David Watts

Help Me Find My IBM @server xSeries Performance Problem!

There are a number of reasons why you need to fix performance problems, and there is usually a cost associated with each of them. To resolve a performance bottleneck problem, you will need to start with three major questions:

- ▶ Where is the bottleneck?
- ▶ How can it be fixed?
- ▶ How much will it cost?

This IBM® Redpaper helps you to detect bottleneck problems on the IBM @server® xSeries® server. It also discusses what to look for in components that are bottlenecks so that you have all the data that you need to identify possible solutions.

The information in this paper is useful if you are facing a situation where a performance problem is already affecting a server. In such a reactive situation, you need to follow a series of steps that you can implement to restore the server to an acceptable performance level. In addition, the experience that you gain over time from solving server bottlenecks can be very useful when doing new server configuration or server consolidation exercises.

You can use the information in this paper as a methodology to help you spot any server performance bottlenecks. We have provided a number of worksheets that you can fill in based on the performance measurements you gather from the server.

Bottleneck-detection strategy

We used the following steps as our bottleneck-detection strategy:

1. Know your system.
2. Determine if the bottleneck is real or simply a misunderstood expectation.
3. Back up the system.
4. Monitor and analyze each subsystem during the time that you expect the bottleneck to occur.
5. Identify the primary bottleneck and any latent bottlenecks.
6. Fix the cause of the bottleneck by trying only one change at a time.
7. Go back to step 4 until you are satisfied with the performance of the system.

Tip: As you follow these steps, you should record the changes you make and their effect on performance.

Bottleneck-detection techniques

You can achieve successful performance tuning by using a variety of techniques. If you want to increase your success rate and reduce the time spent on each case, you should use a repeatable methodology.

- ▶ Perform general maintenance of the server.

The first job is to ensure the server is up-to-date on service packs, patches, and drivers. A good procedure is to do general maintenance on the server: reboot, defragment drives, and get the machine up-to-date on drivers and patches. In some cases, performance bottlenecks are caused by improper service pack, BIOS, driver configuration, or incompatibilities. Ensure the server is up-to-date with the latest system software before you waste time chasing bottlenecks.

- ▶ Develop accurate and realistic goals.

Any performance improvement quest needs to have realistic boundaries. Having unrealistic expectations may lead to spending an unreasonable time and money. Also, do not be afraid to reevaluate your goals during the process.

Research what the expectations are for this server. Does the customer expect the new 8-way server to be twice as “fast” as the older 4-way? This expectation may or may not be valid. When the expectation involves comparing one server to another, check published industry standard benchmarks that are relevant to the production application environment to get a rough idea of how the two servers should compare.

Consider that benchmarks are performed by very skilled engineers who know how to extract the most performance from each system. It might not be possible to obtain a similar result in production environments, so try to be conservative. Also, remember that response times can vary widely, depending upon configuration. Throughput is the only accurate way to compare multi-user capacity of two different servers.

The phrase “our server is slow” can imply several meanings. Will an improvement of 20% be sufficient? This is the perfect time to identify the expectation. If the customer says the server is slow, ask how much faster they need it to be. Document this expectation, but try to obtain a reasonable expectation before you launch into an extensive bottleneck detection effort. If the customer wants the system to be five times faster, that might not be practical without a server replacement, application modification, or both.

- ▶ Gather relevant background information.
This is a key point to successful bottleneck removal. To do so, you should use an iterative process for questioning. A good starting point is using the questions in “Step 1: Gathering information” on page 3.
- ▶ Have a good understanding of the system.
This is half of the solution. Don't try to diagnose a complex multi-server bottleneck if you are just learning the basics. Do your homework and ask for expert assistance if needed.
- ▶ Use a methodological approach.
Time and stress are two frequent parameters during a troubleshooting situation. Going out in the battlefield and “trying this and that” does not get you anywhere. Prepare your battle plan before getting to the site, keep it simple, methodical, and stick with it.
- ▶ List all necessary performance metrics and counters.
Having a good understanding of the system and the performance problems must lead you to a set of parameters that helps you resolve the situation. As you understand the problem in more detail, use more specific counters to focus on the problem. Start with the simple counters first, then dig deeper into the component that is causing the bottleneck.
- ▶ Gather a baseline of the system's current performance.
If you have stated your goals, you need to be able to measure the results of your actions. Without a baseline, you cannot tell if your goals have been met.
- ▶ Validate your interpretation of counter values.
Try, if possible, to record all counters for the recommended objects listed in “Step 2: Monitoring the server's performance” on page 5. This record gives you all the data that you need to drill deeper into problems as you learn more about the bottleneck without having to take multiple traces.
- ▶ Make a permanent record of your progress.
Record your steps, changes, and results in a dedicated performance notebook. Loose paper tends to disappear into the recycling bin. You will thank yourself in six months when similar problems happen. Over time, this notebook will provide useful information to help you build your bottleneck-detection expertise.
- ▶ If in doubt, contact an expert.
Replacing hardware is costly, not only in terms of parts but also in machine downtime. If you do not feel comfortable about a solution, consult experts. When you consult an expert, your good understanding of the system and your notebook will be helpful.

Step 1: Gathering information

Mostly likely, the only first hand information you have are statements such as “there is a problem with the server.” In this situation, you need to ask probing questions of those within your organization to clarify and accurately record the problem. Here is a list of questions you should ask to get a better understanding of the system.

- ▶ Could you give me a complete description of the server in question?
 - Model
 - Age
 - Configuration
 - Peripheral equipment
 - Operating system

- ▶ Can you tell me what the problem is *exactly*?
 - What are the symptoms?
 - Is the number of users on the server now the same as when the server was installed?
 - Describe any error messages.

Some people have problems obtaining this information. Any extra information you find may give you hints about how to solve the problem. For example, someone might say “It is really slow when I copy large files to the server.” This type of performance can indicate a network problem or a disk subsystem problem.

Keep in mind that people often describe problems by discussing poor response time. Knowing server response time is only a part of the picture. The most important metric for predicting server performance is throughput. For example, how many transactions per second or bytes per second is the server sustaining and how much is needed?

Knowing the answers to these questions can help determine if the server will ever be able to support the required load. No amount of server optimization is going to improve performance to the desired level if the customer needs higher bandwidth than the network can possibly sustain.

- ▶ Who is experiencing the problem?

Is one person, one particular group of people, or the entire organization experiencing the problem? Determining who is having the problem will help determine if the problem exists in one particular part of the network, if it is application-dependent, etc. If only one user is experiencing the problem, then the problem might be the user’s PC.

The perception that clients have of the server is usually a key factor. From this point of view, performance problems may not be directly related to the server. The network path between the server and the clients could easily be the cause of the problem. This path includes network devices as well as services provided by other servers such as domain controllers.

- ▶ Can you reproduce the problem?

Most reproducible problems can be solved. If you have sufficient knowledge of the system, you should be able to narrow the problem to its root and decide which actions you should take.

If you can reproduce the problem, you can see the problem and understand it better. Record the sequence of actions necessary to reproduce the problem at any time, using the following guidelines:

- What are the steps to reproduce the problem?

Knowing the steps may let you reproduce the same problem on a different machine under the same conditions. If this process works, you can use a machine in a test environment and remove the chance of crashing the production server.

- Is the problem intermittent?

If the problem is intermittent, the first thing to do is to gather information and find a path to move the problem in the reproducible category. The goal here is to have a scenario to make the problem happen on command.

Important: Knowing whether the problem is intermittent critical. If the problem cannot be reproduced consistently, then (unless you are extremely lucky) there is little chance of finding the bottleneck by taking a trace.

- Does the problem occur at certain times of the day or certain days of the week?
Knowing the time frame of the problem might help you determine what is causing the problem. The problem may occur when everyone arrives for work or returns from lunch. Look for ways to change the timing (that is, make it happen less or more often). If there are ways to do so, it becomes a reproducible problem.
- Is the problem unusual?
If the problem falls into the non-reproducible category, you may conclude it is the result of extraordinary conditions and classify it as fixed. In real life, there is a high percentage that it will happen again.
- ▶ When did the problem start? Was it gradual or did it occur very quickly?
If the performance issue occurred gradually, then it is likely to be a sizing issue, or if it appeared overnight, then a change made to the server or peripherals may be causing the problem.
- ▶ Have any changes been made to the server (minor or major) or are there any changes in the way clients are using the server? Did the customer alter something on the server or peripherals recently that might have caused the problem? Is there a log of all network changes available?
Demands could change based on business changes, which could affect demands on a servers and network systems.
- ▶ Are there any other servers or hardware components involved?
- ▶ Are there any logs available?
- ▶ What is the priority of the problem? When does it need to be fixed?
 - Does it need to be fixed in the next few minutes or days?
 - How massive is the problem?
 - What is the related cost of the problem?

Step 2: Monitoring the server's performance

Important: Before taking any troubleshooting actions, back up all data and the configuration information to prevent a partial or complete loss.

At this point, begin monitoring the server. The simplest way to monitor the server is to run monitoring tools from the server which you are analyzing. For Microsoft® Windows®, the monitoring tool is System Monitor (**Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Performance**).

Create a performance log of the server during its peak time of operation (for example, from 9:00 a.m. to 5:00 p.m.). When creating the log, if available, include a some of the following:

- ▶ Processor
- ▶ System
- ▶ Memory
- ▶ Physical disk
- ▶ Network interface

Based on server type, determine the important subsystems that are likely to be the source of the performance bottleneck. Table 1 lists key subsystems to check on different server types.

Table 1 Server types and important subsystems

Server type	Key subsystems
File servers	<ul style="list-style-type: none"> ▶ Network ▶ Memory ▶ Disk
Print servers	<ul style="list-style-type: none"> ▶ Memory ▶ Disk ▶ Processor
Database servers	<ul style="list-style-type: none"> ▶ Memory ▶ Disk ▶ Processor
E-mail servers	<ul style="list-style-type: none"> ▶ Memory ▶ CPU ▶ Disk ▶ Network
Web servers	Static content: <ul style="list-style-type: none"> ▶ Network ▶ Memory ▶ CPU Dynamic content: <ul style="list-style-type: none"> ▶ Memory ▶ Network ▶ CPU ▶ Disk
Groupware servers	<ul style="list-style-type: none"> ▶ Memory ▶ CPU ▶ Disk I/O
Multimedia server	<ul style="list-style-type: none"> ▶ Network ▶ Memory ▶ Disk
Terminal server	<ul style="list-style-type: none"> ▶ Memory ▶ CPU ▶ Network
Virtualization servers	<ul style="list-style-type: none"> ▶ Memory ▶ Disk I/O ▶ Network

Based on your server type listed in Table 1 on page 6, fill out the information in Table 2.

Table 2 Your server type and key subsystems

Information	Details of your server
Server type	
Important subsystems	1.
	2.
	3.
	4.

Recommended server performance tuning process

Before you begin, remember that a methodical approach to performance tuning is important. We recommend that you use the following process for xSeries server performance tuning:

1. Understand the factors affecting server performance. This paper will help you to do so.
2. Measure the current performance to create a performance baseline that you can compare with your future measurements and that you can use to identify system bottlenecks.
3. Use monitoring tools to identify a performance bottleneck. By following the instructions in the next sections, you should be able to narrow down the bottleneck to the subsystem level.
4. Improve the component causing the bottleneck by performing some actions to improve server performance in response to demands.

Note: The greatest gains are obtained from upgrading a component that is causing a bottleneck when the other components in the server have ample “power” left to sustain an elevated level of performance.

Figure 1 shows the position of five subsystems (CPU, memory, network, disk, and operating system). Each subsystem is represented by a letter from *a* to *e* on a performance scale. A well-optimized system will have all of its subsystems grouped together. This figure has one component (*a*) which represents the system bottleneck. In this case, you need to move component (*a*) closer to the other subsystems.

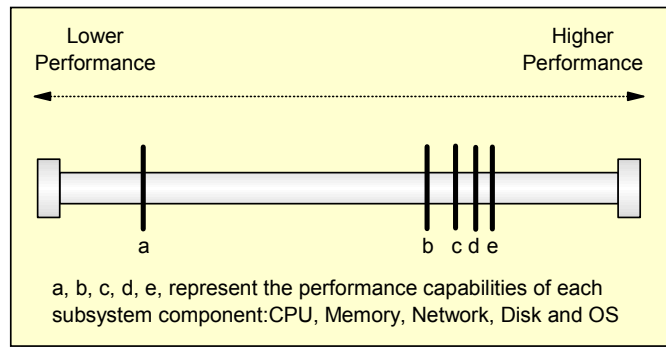


Figure 1 System with one primary bottleneck (*a*) and other well-balanced subsystems

In addition, ensure that other components in the server are not *latent* bottlenecks working just below the utilization of the component that is causing the bottleneck. Latent

bottlenecks will limit improvements that you may see from any upgrade. In general, components that have average utilization of 60% to 70% are likely to be latent bottlenecks.

If there are latent bottlenecks in a system, then you must reconfigure or upgrade both the primary component causing the bottleneck and the component that has a latent bottleneck to obtain optimal performance.

Figure 2 shows one primary (a) and one latent bottleneck (b).

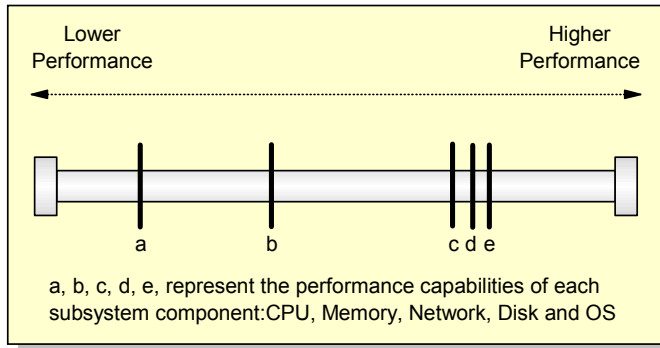


Figure 2 System with a primary (a) and a latent (b) bottlenecks

In this case, upgrading (a) will move the primary bottleneck to (b) as shown in Figure 3. If several components are causing latent bottlenecks, perhaps the most cost-effective solution would be to replace the entire server.

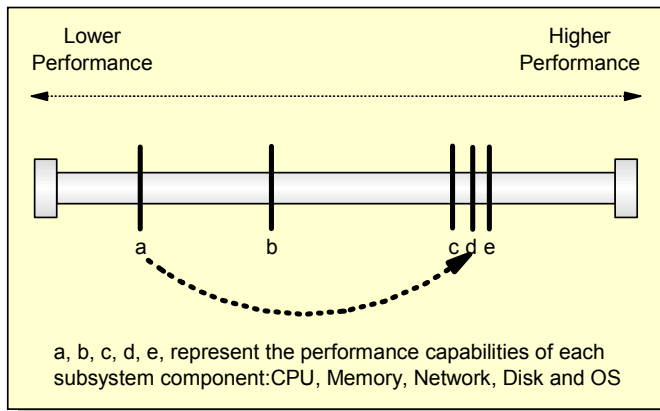


Figure 3 Moving the bottleneck from a to the latent bottleneck (b)

5. Measure the new performance. Measuring will help you compare the performance before and after the tuning steps.

Performance tuning is a continuous process, so you must maintain ample records to help analyze future demands. By doing so, you could foresee and eliminate problems before they even occur.

Analysis tools, such as System Monitor in Windows, give access to all the listed objects and counters in the following tables. How many objects and counters you measure and how often you measure them depends on two questions:

- ▶ What are you trying to detect?
- ▶ What kind of disk resources do you have available?

The sampling period, or the amount of time between data collection points, always comes with a trade off. The smaller the sampling period, the bigger the file will be. On the other hand, an extremely long sampling period is not suited for detecting temporary peaks in the system utilization. For example, if you record all the counters listed in this section, you could expect a log file size for a one second sampling rate to be close to 30 MB per hour of recording.

Tip: For an eight-hour trace, a sample time of 60 seconds or longer is sufficient to diagnose consistently slow server performance.

Keep in mind that the techniques we recommend in this section do not require high-resolution sampling unless the problem you are diagnosing only happens for a brief period of time. If the problem is a consistently slow server, then use sample times that result in trace files of a manageable size.

If space is a problem and you want to record for a long period of time, you should use the circular logging option in the Performance console. Using the alerts function of the Performance console could also help.

Sampling could also be a two-step action. First, record using only core counters from the different subsystems until you can narrow the search to one or two subsystems. After that, increase the number of counters in one subsystem and decrease the sampling rate. The obvious downside to this method is the extra work and inconvenience of taking multiple traces.

Where to start

Now we have a good idea of what we want to accomplish, but how do we proceed? Which counters should we use, and how do we know when a bottleneck occurs? These are the crucial steps which cause many people to stumble, because there are an enormous number of complex performance objects and associated counters. However, detecting server subsystem bottlenecks is quite easy once you understand the primary subsystems and the primary subsystem counters that you can use to diagnose the health of each subsystem.

Our recommended strategy for bottleneck detection involves a *top-down approach*. In a top-down approach, we take a high-level look at each of the server's primary subsystems by examining each of the primary counters which indicate that bottlenecks are occurring.

First, validate the health of each primary server subsystem. The best way to determine if a subsystem is performing poorly is to identify the primary counters and the corresponding thresholds that you can use to identify performance bottlenecks for that subsystem. Then, examine each of the primary counters and compare them to the thresholds to determine if the subsystem is healthy or unhealthy.

These primary performance counters are critical because you can use them as a pass/fail test to determine the health of a subsystem. Only after a subsystem has failed the primary counter test do you need to perform more extensive analysis. This approach makes bottleneck detection much easier because you can avoid all the complex counters for any subsystem that passes the primary counter test.

Table 3 on page 10 lists the primary performance objects and associated counters, along with the corresponding threshold for each of the server subsystems, that we used for our pass/fail test, in order of most likely to cause a bottleneck. Our top-down approach is to perform a pass/fail test for each of the primary counters listed in Table 3 on page 10. This approach provides a simple, objective way to systemically determine if each subsystem is healthy.

Table 3 Primary performance objects

Subsystem	Counter	Guidance	Your value
Disk	Physical Disk: Average Disk sec/Transfer	Must be lower than about 25 ms.	
Memory	Memory: Available Bytes	Should be no lower than 20-25% of installed memory.	
Memory	Memory: Page Reads/sec	Ideally should be zero, but sometimes this is not possible because some applications, such as Lotus® Domino® and SAP, use the page file for memory mapped file communication. In any event, the combined value of this counter and Page/Writes/sec should not be higher than about 150 I/Os per second per disk used in the page file device.	
Memory	Memory: Page Writes/sec	Ideally should be zero, but sometimes this is not possible because some applications such as Lotus Notes® and SAP, use the page file for memory mapped file communication. In any even, the combined value of this counter and Page/Reads/sec should not be higher than about 150 I/Os per second per disk used in the page file device.	
Processor	Processor: % Processor Time_Total	Should be lower than about 70% to 80%.	
Processor	Processor: % Processor Time_(N)	Each processor should be lower than about 70% to 80%.	
Network	Network Interface: Bytes Total/sec	Should be lower than about 50% to 60% of maximum sustainable bandwidth. For Gigabit Ethernet in a Xeon MP or Xeon DP-based system, this is about 70 to 80 MBps. For Gigabit in a Pentium® III-based system, this is about 30 to 40 MBps. Use 1/10 of these values for 100 Mbps Ethernet.	
Network	Network Interface: Packets/sec	Should be no higher than about 70 000 to 80 000 packets per second for Gigabit Ethernet in Xeon MP or Xeon DP-based servers. Should be no higher than about 30 000 to 40 000 packets/sec for Gigabit adapters in Pentium III-based servers. For 100 Mbps Ethernet, use 1/10 of these values.	

Only after we find one or more unhealthy subsystems do we drill down deeper to learn more about the bottleneck. Once we find bottlenecks, we can then determine if they can be relieved by upgrading or tuning the specific subsystem.

After tracing hundreds of server configurations, we have learned that the most likely server hardware components to cause a bottleneck are, in order:

- ▶ Disk subsystem
- ▶ Memory subsystem

Disk and memory technologies have significantly lagged behind the performance curve of processors and network technology, and this is one reason why disk and memory are the two most frequently found server bottlenecks. However, equally important is that many administrators configure server disk subsystems based solely on capacity requirements. Often this means purchasing fewer, higher-capacity disks, which means there are fewer disk heads to service the required I/O data rates. Because the disk subsystem is so often the bottleneck, we will start there.

Disk subsystem

The disk subsystem is comprised of the disk controller, its device driver, the SCSI bus or Fibre Channel bus connecting the system to the disks, and finally, the individual disk drives. One key point to understand about disk subsystems is that for most commercial server workloads, physical disk I/O is almost always random. Servers provide data storage for the entire population of users attached to the network that is connected to that server. In addition, each user is requesting different data from different locations on the disks of the server. Thus, the server is trying to cache data in buffers in memory, and the disk controller is trying to cache data in the controller cache. The OS is also using a file system to store data that will become fragmented over time.

So, by the time a disk I/O actually reaches the disk drive, it is almost always to a very different address than the previous disk I/O request. When this happens, the disk controller has to process the I/O command, send the command to the disk, move the head to the new data track (seek), wait for data to rotate underneath the head (rotational latency), read or write the data, and then send the completion status back to the controller to notify the OS that the I/O is complete.

Even if users on the server are executing applications that are performing sequential I/O, because of all the caching and disk fragmentation, much of the physical I/O at the disk drive can still be random. Also, most commercial applications do not perform sequential I/O. Databases, e-mail, file serving, and most multi-user commercial applications generate random disk I/O which introduces seek and rotational latency delays for nearly every disk access, thus greatly reducing sustained throughput. In such cases, we should not expect disks to generate very high data rates for most commercial servers.

Of course, there are always exceptions. For example, high performance computing (HPC) servers might run a single process that reads a large array of data from disk and then writes a large solution set of data back to disk. In this case, a single process accesses the disk, instead of a large number of concurrent users generating multiple, unique disk I/O requests. Furthermore, HPC workloads tend to read disk data using very large disk I/O sizes, thereby increasing the sustained disk I/O bandwidth.

For HPC workloads, it is possible to saturate PCI, SCSI, or Fibre Channel bandwidth limitations, but in most commercial workloads, seek and latency operations dominate the I/O time and significantly lower sustained I/O rates.

A closer look at the time to perform disk I/O operations provides a critical understanding of how to avoid and diagnose disk subsystem bottlenecks for commercial workloads (random I/O).

Let's take a look at a typical high-speed 15 000 RPM disk drive. To calculate the total access time for this disk, add the time values for average seek, average latency, and command and data transfer as illustrated in the following equation:

Average seek	3.8 ms
Average latency	2.0 ms
Command and data transfer	< 1 ms
<hr/>	
Average random access time	6.8 ms per operation (147 operations/sec)

In this case, it takes about 0.0068 seconds for a 15 000 RPM disk to do an average disk operation. Therefore, in one second the disk can only do about 147 I/Os per second. This is calculated as:

$$1 / 0.0068 \text{ sec} = 147 \text{ I/Os per second per disk.}$$

Because most commercial applications are accessing data on disk in 4, 8, or 16 KB sizes, the average bandwidth sustained by a disk drive can be calculated easily. For example, at 8 KB I/O size:

$$8 \text{ KB per I/O} * 147 \text{ I/Os per second} = 1.15 \text{ MBps per disk}$$

At about 1.15 MBps per disk, it takes a *lot* of disks to start to stress the PCI bus (a 100 MHz PCI-X can handle 800 MBps), the SCSI bus (Ultra 320 is 320 MBps), or Fibre Channel (2Gbit = 200 MBps full duplex). Yet, far too many people worry about PCI bus and SCSI bus configuration, when in fact these are usually the last thing to worry about.

In general, the number of disk drives, disk fragmentation, RAID strategy, and the ability of the application to queue a large number of disk I/O commands to the physical array are the leading causes of commercial server disk subsystem bottlenecks.

The total access time for a 10 000 RPM disk drive would be as follows:

Average seek	4.9 ms
Average latency	3.0 ms
Command and data transfer	< 1 ms
<hr/>	
Average random access time	8.9 ms per operation (112 operations/sec)

8.9 ms corresponds to $1 / 0.0089 = 112$ I/O operations per second, and at 8 KB per I/O, a 10 000 RPM disk can sustain only about:

$$8 \text{ KB per I/O} * 112 \text{ I/Os per second} = 896 \text{ KBps per disk} (\approx 900 \text{ KBs})$$

Now, it might be tempting to use the I/O rates that we just calculated as indicators for disk performance bottlenecks, but this would be a grave mistake because I/O rates can vary wildly, for example:

- ▶ In some special cases, disks will perform sequential I/O. When this situation occurs, seek and rotational latency will be zero or near-zero, and disk I/O rates will increase dramatically. Even though this is rare, we don't want to use a performance indicator that works only some of the time.
- ▶ Our calculations assume average seek and latency times. Drive vendors produce average seek times from 1/3 track-seek range measurements. Full track seek times are much longer, and disks that are accessing more than 1/3 of capacity will have longer seek times and significantly lower sustained I/O rates.
- ▶ RAID strategy affects the number of physical I/Os a disk will actually perform. A random write to a RAID-5 disk array will generally produce two read and two write disk operations to the RAID-5 array. Yet, operating system disk counters will count this as one disk I/O.
- ▶ Stripe size and I/O request size will affect the number of physical disk operations performed. For example, a very large disk read or write operation for 64 KB in size sent to an array that is using 4 KB stripe size will generate 16 physical disk I/O operations. But the OS disk counters will count this as one disk I/O because the OS does not know anything about the stripe size that the disk array controller is using.

Clearly sustained disk I/O rates can vary greatly. So we *don't* recommend using average disk I/O rates as an indicator of a disk bottleneck unless you thoroughly understand the disk workload and storage configuration.

A better way to identify disk bottlenecks is to use our understanding of disk operation combined with average response time. We know that a 15 000 RPM disk requires about 7 ms of average disk access time and that a 10 000 RPM disk has about 9 ms of disk access time.

We can use this information to greatly simplify disk bottleneck detection. But before we launch into that discussion, we need to discuss one more characteristic of disk drive operation: optimization.

Modern disk drives can actually increase the sustained throughput when given more work. If multiple read and write operations are sent to a disk drive, it can use *elevator seek optimization* and *rotational positioning optimization* to reorder the physical I/Os to increase the sustained I/O rate as opposed to processing a single disk read or write operation at a time.

By sending two or more disk commands to the disk, the disk can reorder the operations to reduce the amount of seek time and even rotational latency. But even more significantly, when a seek is occurring for an existing I/O request and another disk I/O command arrives, the disk can determine if it can access that data while performing the current seek command. That is, while a long seek operation is occurring, the processor on the disk determines if the head will pass over any of the data addresses for read or write commands that just arrived in its queue after the long seek was started. If so, the read or write command in the queue will be run while the head is moving out to the track for the original I/O.

Key message: Disk drives perform best and have optimal throughput when given two to three (but no more than three) disk operations at the same time.

As a rule of thumb, the optimal response time of a disk drive is about 2.5 times the normal access time.

- ▶ For 15 000 RPM disk drives this is about 17 ms.
- ▶ For 10 000 RPM disk drives the optimal response time is about 22 ms.

For bottleneck detection purposes, use a range of values instead of a precise number. A good rule of thumb is that a disk subsystem is healthy whenever the disk subsystem is performing read and write operations with less than about 20 to 25 ms per I/O. When the average disk latency is much greater than 25 ms, then the disk subsystem can be considered unhealthy and is a bottleneck.

Rule of thumb: When Avg. Disk Seconds/Transfer (the disk latency counter) is significantly greater than 25 ms, the disk subsystem is unhealthy and is a bottleneck. Remember, this counter does not tell us *how* to fix the problem, it only indicates there *is* a problem.

We can look at one simple Performance Monitor counter and know if our disk subsystem is healthy or not. Simply look at Avg. Disk sec/Transfer for each physical disk drive or array in your server. Use chart mode to identify the peaks, and if this counter spends a significant amount of time over 25 ms during the period where the server is considered to have a bottleneck, you can consider your disk subsystem unhealthy.

Do not order a new SAN if your average disk latency is 26 ms. Clearly there is a range of latencies where performance will be acceptable. Each server administrator must determine when to consider the average latency too great. Some server administrators will use 30, 40, or 50 ms, others will want the ultimate performance and will take action at 25 ms.

However, if the disk subsystem is running at 60 or 80 ms on a regular basis, then the disk subsystem is clearly slowing down server performance. On many occasions, we have seen overloaded disk subsystems performing with one or two seconds of average latency. This computes to 1000 or 2000 ms and is a very slow disk subsystem.

When performing clustering, the threshold is a little higher. Clustering solutions with SCSI require the disk subsystem to disable write-back mode. When RAID-5 is used in write-through mode, the server must perform two reads and two writes for each and every write command. In this case, writes can take twice as long as RAID-0 or RAID-1. So, for clustering solutions or in any case where the server is performing a large amount of write operations for RAID-5 with write-through disk controller settings, you should use about 40 to 50 ms as the threshold for identifying disk performance bottlenecks.

Complete Table 4 with your results. When you are done, examine the primary disk counter in the table to determine if the disk subsystem is a bottleneck.

Table 4 Performance console counters for detecting disk bottlenecks

Counter	Definition and optimal values	Your result
Physical Disk: Avg. Disk sec/Transfer	This is the primary counter for detecting disk bottlenecks. The time to complete a disk I/O. For optimal performance, this should be less than 25 ms. Consistently running over .025 (25 ms) indicates disk congestion. Note: Examine the counters for physical disk <i>not</i> logical disk.	
Physical Disk: Avg. Disk Queue Length	This counter is useful to determine how many disks to use in a RAID array. In general, you obtain optimal performance when this counter averages two to three times the number of disks in each physical array. A high number indicates queuing at the physical volume which increases response time and degrades performance. However, a high number also can be good, because it indicates the application I/O workload will scale simply by adding disks to the array.	
Physical Disk: Avg. Disk Bytes/Transfer	The average number of bytes transferred to or from the disk during write or read operations. You should also compare this value against the stripe size of the RAID array (if you are using hardware-based RAID). We recommend you configure the stripe size to be at least equal to the long-term average value of this counter. For example, if the Avg. Disk Bytes/Transfer is 4 KB, then use an 8 KB stripe size on the RAID array volume.	
Physical Disk: Disk Bytes/sec	Sum this counter's value for each disk drive attached to the same SCSI/Fibre Channel controller and compare it to 70% to 80% of the theoretical throughput. If these two numbers are close, the bus is becoming the disk subsystem's bottleneck. Review the disk subsystem data path.	
Physical Disk: Split IO/sec	A split I/O is a result of two different situations: the data requested is too large to fit in one I/O, or the disk is fragmented.	

Memory subsystem

Most memory counters are related to virtual memory management. Virtual memory counters will not help us identify bottlenecks if our server has insufficient physical memory capacity and is running poorly as a result of excessive disk paging. However, there are a few counters that can determine if the memory configuration is healthy.

Complete Table 5 on page 15 by monitoring the memory counters on your server. Then, examine the primary memory counters to determine if the memory capacity is causing a bottleneck.

Table 5 Performance console counters for detecting memory bottlenecks

Counter	Definition and optimal values	Your result
Memory: Page Reads/sec	<p>This is a primary memory bottleneck counter.</p> <p>Ideally this value should be close to zero. However, sometimes it is not possible to eliminate paging, because some applications (such as Lotus Domino) use the page file for communication between processes.</p> <p>However, if paging is so high that it saturates the page disk device then performance will suffer. If this counter is consistently higher than 150 I/Os per disk per second for the paging device, the server has a memory or paging device bottleneck.</p> <p>Page Reads/sec is the rate at which the disk was read to resolve hard page faults. It shows the number of read operations, without regard to the number of pages retrieved in each operation. Hard page faults occur when a process references a page in virtual memory that is not in a working set, or elsewhere in physical memory, and must be retrieved from disk.</p> <p>This counter is a primary indicator of the kinds of faults that cause system-wide delays. It includes read operations to satisfy faults in the file system cache (usually requested by applications) and in non-cached mapped memory files. Compare the value of Memory:Pages Reads/sec to the value of Memory:Pages Input/sec to determine the average number of pages read during each operation.</p>	
Memory: Page Writes/sec	<p>This is a primary memory bottleneck counter.</p> <p>Ideally this value should be close to zero. But sometimes this is not possible to eliminate paging because some applications use the page file for communication between processes. However, if paging is so high as to saturate the page disk device then performance will suffer. If this counter is consistently higher than 150 I/Os per disk per second for the paging device, the server has a memory or paging device bottleneck.</p> <p>Page Writes/sec is the rate at which pages are written to disk to free up space in physical memory. Pages are written to disk only if they are changed while in physical memory, so they are likely to hold data, not code. This counter shows write operations, without regard to the number of pages written in each operation. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.</p>	
Memory: Available megabytes	<p>This is a primary memory bottleneck counter.</p> <p>If this value stays at less than 20% to 25% of installed RAM consider adding memory to the server.</p>	
Memory: Pool Nonpaged Bytes	<p>Indicates the amount of RAM in the non-paged pool system memory area where space is acquired by operating system components as they accomplish their tasks. If this value shows a steady increase without a corresponding increase in activity on the server, it may indicate that a process that is running has a memory leak, and it should be monitored closely.</p>	
Paging File: % Usage Peak	<p>This is a bottleneck if this value consistently reaches 90%.</p>	
Server: Pool Nonpaged Peek	<p>The maximum number of bytes of nonpaged pool the server has had in use at any one point. Indicates how much physical memory the computer should have. Add 20% to this value to determine the amount of installed memory that the server should require.</p>	
Server: Pool Nonpaged Failures	<p>The number of times allocations from nonpaged pool have failed. Indicates that the computer's physical memory is too small. If this value is non zero on a regular basis, the system needs more memory.</p>	

Processor subsystem

Determining processor bottlenecks is easy. If the %Processor Time Total, or any single processor is sustaining over 70% to 80% utilization it should be considered a bottleneck. Complete Table 6, then examine the primary processor counters to determine if the CPU is a bottleneck.

Table 6 Performance console counters for detecting CPU bottlenecks

Counter	Definition and optimal values	Your result
Processor:% Processor Time	<p>This is the primary counter for detecting processor bottlenecks.</p> <p>The CPU is a bottleneck if the value is consistently running over 70% to 80% (excluding any processes that are running in the background in low priority absorbing all spare CPU cycles).</p> <p>Note: Examine the counters for each CPU installed as well as the _Total value to ensure there are no problems with “unbalanced” or processor affinitized applications.</p>	
Processor: % Privileged Time	<p>This counter can be used (excluding any processes that are running in the background in low priority absorbing all spare CPU cycles) to identify abnormally high kernel time and may indicate I/O driver problems.</p> <p>Note: Examine the counters for each CPU installed as well as the _Total value to ensure there are no problems with “unbalanced” applications.</p>	
Processor: % User Time	<p>%User time represents the time spent by the user application on the server. This is an important counter because it shows a breakdown of how the server application is utilizing all the processors.</p> <p>Note: Examine the counters for each CPU installed as well as the _Total value to ensure there are no problems with “unbalanced” application usage. Often, when applications do not scale, they will not start user threads for all the processors in the server. Noticing processors 0, 1 running high %User Time while processors 2, 3, etc., are running much lower %User Time indicates insufficient application threading.</p>	
System: Processor Queue Length	<p>A sustained queue of much more than four times the number of processors installed indicates a processor congestion.</p> <p>The good news is that an application that drives the processor queue to a long length can take advantage of a large number of processors.</p> <p>When the server is running at above 75% CPU utilization, check this counter to see if the average queue length is significantly greater than two times the number of installed processors. If so, that application will scale as additional processors are added, up to the point where the Avg. Queue Length is equal to 2N, where N is the number of processors.</p> <p>You always want to see at least two threads per processor. Ideally, twice the number of processors is the optimal queue length when Hyper-Threading is enabled.</p>	
Processor Interrupts/sec	<p>Processor Interrupts/sec should no longer be used as a simple indicator of performance.</p> <p>Modern device drivers have dynamic mechanisms and batch interrupts when the system is busy, doing more work per interrupt. This causes the number of interrupts per second to be dynamic.</p> <p>So, when a system is moderately busy, it might require an interrupt to process each LAN or DISK I/O operation. In this mode, the server will have a fairly high interrupt rate. However, as the server becomes busier, multiple disk and LAN operations will be sent under one interrupt request, lowering the interrupt rate and improving interrupt efficiency.</p> <p>In summary, do not use this counter unless you have detailed information about the specific device drivers used in your server.</p>	No need to measure, because this counter is not normally used to analyze bottlenecks.

Network subsystem

The network itself can be difficult to analyze. This is because the performance counters captured on the server represent only the load in the network to or from the particular server where the counters were captured. The counters do *not* reflect the entire load in the network which could be causing a serious bottleneck for users connected to the same network as they try to communicate with the server.

Unfortunately, the only way to identify network bottlenecks which are not related to the server where the performance counters were monitored is to use a network analyzer. Using a network analyzer is beyond the scope of this publication.

However, you can use the network subsystem counters to successfully diagnose network bottlenecks caused by excessive traffic to or from a particular server. This excessive traffic will manifest itself as a choke point at the network adapter. Network adapters can have two types of bottlenecks.

- ▶ When the sustainable throughput of the network adapter is reached.
- ▶ When the maximum sustainable packet rate of the network adapter is reached.

In general, these values will vary with network adapter type and system configuration, so they values should not be considered absolute. However, the counter Bytes Total per second should be lower than about 50% to 60% of maximum sustainable bandwidth, meaning the following values:

- ▶ For Gigabit Ethernet on a Xeon MP/DP-based server: 70 to 80 MBps
- ▶ For 100 Mbps Ethernet on a Xeon MP/DP-based server: 7 to 8 MBps
- ▶ For Gigabit Ethernet in Pentium III-based systems: 30 to 40 MBps
- ▶ For 100 Mbps Ethernet in Pentium III-based systems: 3 to 4 MBps

Packets per second rates should be no higher than:

- ▶ For Gigabit Ethernet on a Xeon MP/DP-based server: 70 000 to 80 000 pkts/sec
- ▶ For 100 Mbps Ethernet on a Xeon MP/DP-based server: 7000 to 8000 pkts/sec
- ▶ For Gigabit Ethernet in Pentium III-based systems: 30 000 to 40 000 pkts/sec
- ▶ For 100 Mbps Ethernet in Pentium III-based systems: 3000 to 4000 pkts/sec

Complete all the fields in Table 7 on page 18. Then, compare the primary network counters with the sustained thresholds to determine if the network subsystem is the bottleneck.

Note: In Windows 2000, you will need to load the Network Monitor Driver and SNMP services, as described below, before you can perform this analysis.

To monitor network-specific objects in Windows 2000, you need to install the Network Monitor Driver. (This is not necessary for Windows 2003.) To install the Network Monitor Driver, follow these steps:

1. Open Network and Dial-up Connections in Control Panel.
2. Select any connection.
3. Click **File** → **Properties**.
4. In the General tab, click **Install**.
5. Select **Protocol**.
6. Click **Add**.
7. Select **Network Monitor Driver**.
8. Click **OK** and then click **Close**.

Note: As per Microsoft Knowledge Base entry Q253790, the Network Segment object has been removed from Windows 2000.

Table 7 Performance console counters for detecting network bottlenecks

Counter	Definition and optimal values	Your result
Network Interface: Bytes Total/sec	This is a network subsystem primary counter. Sustained values over 50% to 60% of the adapter's available bandwidth indicates a bottleneck. So for Gigabit Ethernet in a Xeon MP/DP-based system, this is about 70 to 80 MBps For Gigabit Ethernet in Pentium III-based systems, this is about 30 to 40 MBps. Use 10% of these values for 100 Mbps Ethernet.	
Network Interface: Bytes Received/sec	This is a network subsystem primary counter. Sustained values over 50% to 60% of maximum sustained throughput in the receive direction should be investigated by a network administrator to determine if the network is a bottleneck. Most Gigabit Ethernet adapters can sustain about 800 Mbps in the receive direction. Maximum sustained throughput for 100 Mbps Ethernet in the receive direction is about 80 Mbps.	
Network Interface: Bytes Sent/sec	This is a network subsystem primary counter. Sustained values over 50% to 60% of maximum sustained throughput in the send direction should be investigated by a network administrator to determine if the network is a bottleneck. Most Gigabit Ethernet adapters can sustain about 800 Mbps in the send direction. Maximum sustained throughput for 100 Mbps Ethernet in the send direction is about 80 Mbps.	
Network Interface: Packets/sec <i>and</i> Network Interface: Packets Sent/sec <i>and</i> Network Interface: Packets Received/sec	These are network subsystem primary counters. Packets/sec rates should be no higher than about 50% to 60% of maximum packet/sec rates listed in Table 8 on page 18 and Table 9 on page 19. Use the Bytes/Total/sec counter value divided by Total Packet/sec counter value to calculate average Bytes per Packet or average packet size for your server workload. Then using the calculated average packet size (bytes/packet), use Table 8 on page 18 and Table 9 on page 19 to determine the maximum packet per second rate sustainable for the adapter speed being used. We recommend detailed network analysis if the sustained values are 50% or greater than the values listed in the table.	

The following tables are for throughput in packets per second for both 100 Mbps Ethernet (Table 8) and Gigabit Ethernet (Table 9 on page 19). These are reasonable peak rates you can expect from leading server network adapters. Expect much lower throughput if you use less expensive client adapters in your server.

Table 8 100 Mbps Ethernet — expected maximum (peak) packet rates for given packet sizes

Average packet size (bytes)	100% server receives (packets/sec)	70/30 send/receive (packets/sec)	100% server sends (packets/sec)
64	105,790	103,014	126,553
128	89,939	90,661	90,428
256	55,897	67,044	56,140
512	31,789	39,994	31,788
1024	17,028	21,701	17,024
1460	12,206	15,610	12,202

Average packet size (bytes)	100% server receives (packets/sec)	70/30 send/receive (packets/sec)	100% server sends (packets/sec)
2 KB	17,021	23,871	17,024
4 KB	12,998	17,766	13,001
8 KB	12,999	18,278	13,002
16 KB	12,999	18,238	13,004

Table 9 Gigabit Ethernet — expected maximum (peak) packet rates for given packet sizes

Average packet size (bytes)	100% server receives (packets/sec)	70/30 send/receive (packets/sec)	100% server sends (packets/sec)
64	125,780	169,218	146,017
128	164,597	161,256	140,588
256	153,649	158,360	137,616
512	135,439	146,796	151,588
1024	114,985	120,586	158,479
1460	107,760	114,046	120,701
2 KB	122,901	157,194	165,803
4 KB	124,334	151,746	123,718
8 KB	102,238	157,377	126,362
16 KB	115,153	158,683	98,024

Step 3: Fixing the bottleneck

Once you determine which subsystem is the bottleneck, you should examine the options for solving the problem. Depending on your specific situation, these options could include:

- ▶ CPU bottleneck:
 - Add more processors.
 - Switch to processors with larger L2 cache.
 - Replace existing processors with faster ones.
- ▶ Memory bottleneck:
 - Add memory.
- ▶ Disk bottleneck:
 - Spread the I/O activity across drives or RAID arrays (logs, page file, etc.).
 - Add disks to the RAID array.
 - Use RAID-1 instead of RAID-5 or instead of single disks.
 - Correct the stripe size used to match the I/O transfer size.
 - Use faster disks.
 - Add another RAID controller/channel or Fibre Channel host adapter.

- For Fibre Channel, add a second RAID controller module.
- If running in Write Back (WB) mode, then select Write Through (WT) as a temporary fix if additional drives are not available. Selecting WT whenever Avg. Disk sec/Transfer exceeds 15 to 18 ms can yield a 20% to 30% increase in throughput for heavily loaded disk configurations compared to WB mode.
- ▶ Network bottleneck:
 - Ensure network card configuration matches router and switch configurations (for example, frame size).
 - Modify how your subnets are organized.
 - Use faster network cards.
 - Add network cards.

When attempting to fix a performance problem, remember the following:

- ▶ Take measurements before you upgrade or modify anything so that you can tell if the change had any effect (that is, take baseline measurements).
- ▶ Examine the options that involve reconfiguring existing hardware, not just those that involve adding new hardware.
- ▶ Once you upgrade a specific subsystem, other latent bottlenecks may appear in other subsystems.

Figure 4 on page 21 shows a bottleneck flowchart.

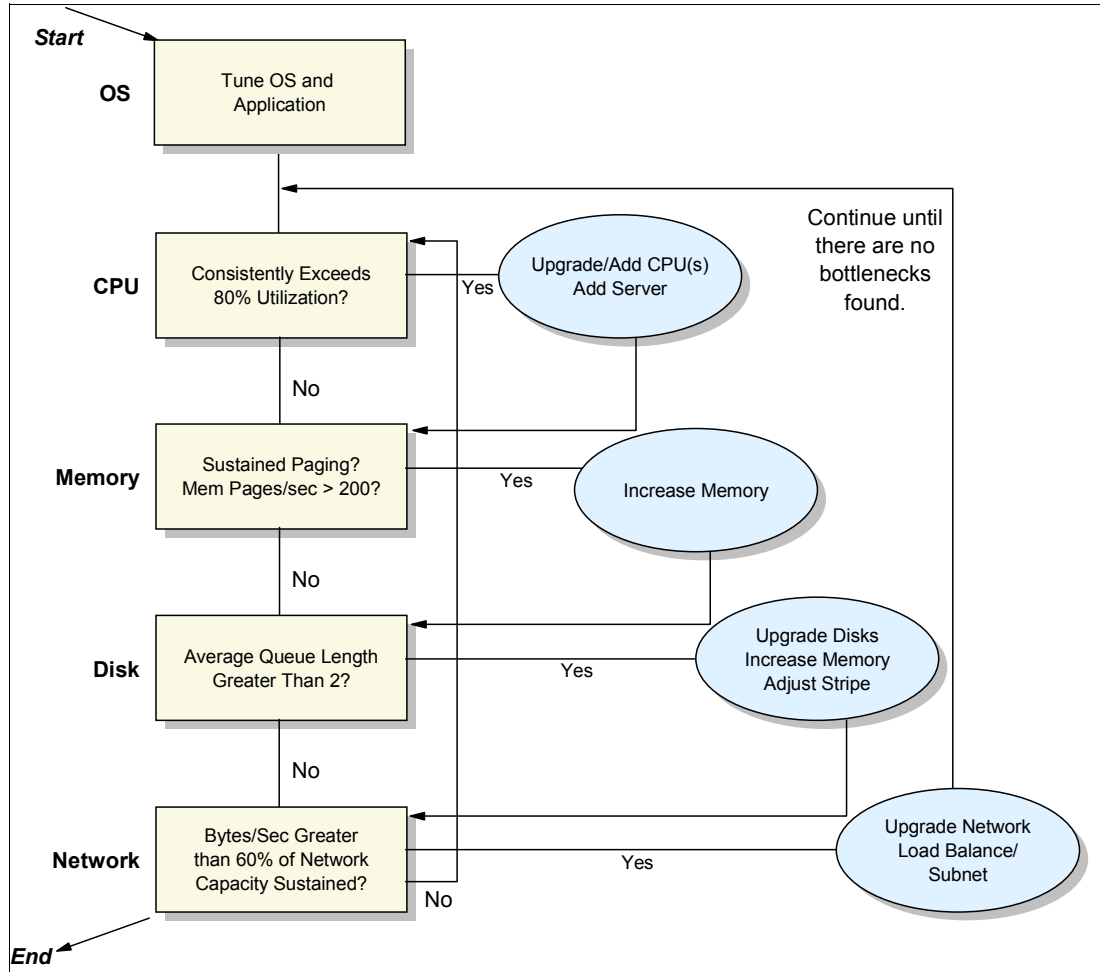


Figure 4 Bottleneck flowchart

Conclusion

This Redpaper provides a general approach to determining server bottlenecks. When trying to find bottlenecks, you should also take into consideration the type of server you are monitoring and what subsystems are potential bottlenecks.

Before making any recommendation for a server:

- ▶ Make sure you understand what is causing the bottleneck.
- ▶ Research your recommendations, and be sure what you are proposing will actually improve server performance.
- ▶ Know how much the upgrade/reconfiguration will cost.

It is also good practice to install IBM Director with Capacity Manager for ongoing and proactive analysis of the server.

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



Gregg McKnight is an IBM Distinguished Engineer and is the technical lead for the xSeries Server Development and Performance organization in Research Triangle Park in North Carolina. He is an Electrical Engineer and has 20 years of experience in performance-related activities on IBM Servers. The Server Development and Performance organization is responsible for server simulation, design and performance optimization, as well as developing key industry benchmarks such as TPC-C, TPC-H, TCP-W, SAP, Notesbench, SPECWeb, NetBench, ServerBench, and WebBench benchmarks. Gregg created the algorithms for rules-based performance analysis that are used in the Capacity Manager extension of IBM Director and is the author of the IBM white paper *Fundamentals of Server Performance*.



David Watts is a Consulting IT Specialist at the IBM ITSO Center in Raleigh. He manages residencies and produces Redbooks™ on hardware and software topics related to xSeries systems and associated client platforms. He has authored over 30 Redbooks and Redpapers. He has a Bachelor of Engineering degree from the University of Queensland (Australia) and has worked for IBM for over 15 years. He is an Certified Specialist for xSeries and an IBM Certified IT Specialist.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

This document created or updated on October 26, 2004.




Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
redbook@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®
@server®
Redbooks (logo) ™
ibm.com®

xSeries®
Domino®
IBM®
Lotus Notes®

Lotus®
Notes®
Redbooks™

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.