Measuring and Monitoring NT Performance

Jerry L. Rosenberg
SRM Associates, Ltd.

NT offers a wealth of measurement data regarding system, interface and application performance. In fact, there is so much information that the greatest danger may well be data overload. This presentation will offer a set of key metrics that will quickly supply an overview of system performance to the analyst. These can be used to quickly assess the status of an NT system, identify bottlenecks and select appropriate corrective actions. A process will be outlined to monitor these metrics regularly to proactively identify and avoid impending bottlenecks.

Background

There is a very rich set of performance data available in NT environments including Windows 2000. Many performance counters are trackable with the standard, supplied performance-monitoring tool (PerfMon in NT and SysMon in W2K). The volume of potential data can turn into a serious management problem that can overwhelm the analyst if a process is not implemented for quickly identifying performance issues and determining the proper approach to addressing existing or emerging bottlenecks. The problem is further exacerbated by the similar counter names that exist across multiple objects. Remember, information is not knowledge and data, especially raw data, is not even information until it is analyzed and understood. What we need to begin with is a methodology to guide our investigation of system performance. One which helps us limit the scope of the initial investigation to allow for a quick review of performance metrics and an assessment of further actions to be taken.

Basics

Performance analysts have been looking at system data for over 30 years and the basics have not changed significantly. A server can only be engaged in three specific types of activity:

- It can be acquiring or archiving information to be processed. This can be either data or instructions. This is I/O.
- It can be holding this data in its internal storage in some manner. This is memory.
- It can be performing operations (instructions) on the stored information (data) . This is CPU or Processor.

This defines the three types of resource that can be utilized in a computing system. If you are thinking that I have forgotten Network, we will cover it in just a minute.

For now, let's agree that there are only three components to any computer system - CPU, I/O and memory.

Over the years I have found it necessary to reevaluate this basic truth and I find that it has withstood the test of time and technological innovation. Periodically, I consider such items as cache controllers and expanded storage and debate as to whether they are memory or I/O or some hybrid of both. Adding a little complexity is the existence of the network component. I like to think of this as another form of I/O subsystem. I have found it best to logically consider the network as basically another transport mechanism for information. In short, specialized I/O. It is subject to the same requirements of path availability and server utilization that we apply to DASD subsystems. But since it has become such a key component of service delivery, it deserves special consideration and discussion of the specific metrics that are available.

By and large, I have come back to the realization that all computer work generates CPU usage, is fed by input and creates output and takes up some amount of storage. As such, the understanding of computer performance requires an awareness of how these three commodities are utilized and the symbiotic effects of each on the others. This leads to the second step of the process - the CPU-I/O-memory interdependency. This relationship allows the analyst to focus corrective action by isolating the root cause of the problem.

1. If the CPU is at 100% utilization or less and the required work is being completed on time, everything is okay for now. (But always remember tomorrow is another day).
2. If the CPU is at 100% busy and all the required work is not completed, you have a problem. Begin looking at the CPU level.
3. If the CPU is not 100% busy, and all work is not completed, a problem also exists and the I/O and memory subsystems should be investigated.

At first look this seems a trifle simplistic. For example, rule 2 could take place if a problem existed in the memory subsystem causing excessive CPU due to paging activity. While this is a memory problem, the way to uncover it is to decompose CPU utilization into its component parts and to see which of these is excessive. In this example, excessive CPU on behalf of paging would suggest an investigation of memory usage. Additionally, rule 3 will generally manifest itself as a result of queuing for I/O or memory or long waits in either subsystem.

This would appear to supply performance analysts with a ”cookbook” approach to investigating computer systems. With NT systems, there are some specific issues that one must consider. While the CPU is still the easiest resource to report and is the measurement element that management will most likely want to see graphed, NT’s voracious appetite for memory gives us a few new observations to consider:

1. NT servers will run short of memory before any other resource. Watch this carefully. NT seems to consume memory.
2. Even when it looks like there is another resource problem, check memory as well.
3. When memory problems have been ruled out, disk and network should be checked next.
4. The best performance gains in NT will come from tuning memory, disk and network subsystems, in that order.
5. In case it hasn’t become obvious, NT exhibits a change from previous platforms in that CPU is the least important resource to consider. This does not mean that you can never have a CPU problem or that the processor speed and quantity will never be insufficient to your workload, but rather that day-to-day performance problems will likely be in the other resources.

In order to insure that we look in the right place first, we should develop a systematic approach to

performance management. Here are some ground rules for the monitoring and tuning process:

1. Start monitoring your system now and develop a baseline.
   It is imperative that you know how your servers perform under normal conditions. It is useful to apply rules of thumb, but your systems may be unique and you should find out if this the case. A good place to start is by collecting the data suggested in this paper over time and reviewing the performance characteristics when there are no user complaints. This will give you a foundation for readily comparing problem periods with acceptable periods in determining where the problem might lie. A second approach is to develop benchmarks of specific conditions to predict the performance of the system under varying workloads or hardware conditions, a valid baseline is critical to the development of an accurate benchmark scenario.
2. Monitor and review your system regularly to anticipate problem.
   By tracking the metrics outlined below, you will be able to anticipate problems by observing changes in the utilization patterns of key resources. These changes should trigger investigation and correction of emerging problems prior to chaos.
3. Identify the problem resource using the program outlined above; using the metrics defined below.
4. Make sure you have a tested backup of the system, files and applications before making any changes.
5. Whenever possible, change one variable at a time and document everything.
6. Benchmark after the change to quantify the effects and to ensure system stability.
7. Go back to step 2.

Collecting data

Now we can begin the actual collection of data. Well, almost. First, there are three things to consider and implement if required.

1. If you are interested in network data, make sure to add the SNMP service to collect data on the network interface object so that you will get info concerning the network interface card.
   Select Start/Settings/Control panel/Networks/Services/Add and add the SNMP service. You will need to reboot to activate.
2. You should also add the Network Tools and Agent to collect network segment data.
   Select Start/Settings/Control panel/Networks/Services/Add and add Network Tools and Agent. You will need to reboot to activate. Note: network monitoring under NT 4.0 will put the selected NIC in promiscuous mode, which will add overhead. Not so in W2K.
3. I strongly urge you to collect disk performance statistics by turning on diskperf. The rumors of high overhead are unfounded on today's processors. Just enter diskperf –ye from the command prompt and reboot. Full options for the diskperf command can be obtained by typing diskperf /? At the command line. This facility will stay on through subsequent reboots until it is disabled.

And now, finally, a look at the metrics that we should be collecting. Remember, we are defining the objects and counters that constitute the starter set to assist in obtaining a baseline understanding of system performance and will alert you to changes in behavior that foreshadow resource constraints. These are a foundation set of metrics. There are many other

counters available that can be employed to drill down for a more detailed evaluation of specific resource behavior. We cannot discuss all of them in the time allotted, but the analyst should make use of the references supplied to develop a second level performance process after a baseline understanding of system behavior has been developed with the metrics discussed here. Please note that the conventions I have adopted for representing metrics is to give the object name followed by a colon and then, the counter name.

Monitoring Processor Usage

The following group of counters will assist in evaluating and tuning processor usage. They all represent the percentage of elapsed time that the processor is executing the non-idle thread in the particular mode of interest. Note: In pre-W2K systems, these fields can also be found in the System object.

Processor: % Processor Time (total instances)

This is the average usage across all processors in the server. Should be compared with the values of the next set to insure balance across processors.

Processor: % Processor Time

This set of counters will allow you to monitor the individual usage at each processor. With Symmetric Multiprocessing, individual processor statistics should not vary significantly from each other.

Values in the above two counters should be investigated further if increasing values (or, generally, values consistently > 80%) are accompanied by Queue Lengths of greater than 2 per processor. These indicate that the processor is becoming a bottleneck.

Processor time is further divided into the following counters that should be investigated if the CPU is considered a bottleneck:

Processor: % User Time

If this value is greater than Privileged Time, then the best results will come from tuning user/application processes.

Processor: % Privileged Time

If this value is greater than User Time, then concentrate on tuning the use of system resources.

Processor: % Interrupt Time

If this counter is > 20% and rising compared to the baseline, check out peripherals and drivers to insure that they are functioning properly.

It should be noted that the same usage counters for User, Privileged and Interrupt (also DPC which is included in Privileged) can be found in the Process and Thread objects. This allows the analyst to drill down to the process and thread levels to determine which specific tasks are consuming CPU.

Other counters that should be reviewed for processor usage are:

System: Processor Queue Length

Probable bottleneck when > 3 * # CPUs. View this value in combination with the processor usage counters.

Processor :Interrupts/sec

Hardware devices use interrupts when they require attention. . Watch for sudden increases when there is no corresponding increase in workload; this could be an indication of a malfunctioning device or device driver.

System: Context Switches/sec

Context switches indicate switches from one thread to another. It is the result of a running thread voluntarily relinquishing control, preemption by a higher priority thread or switches between user mode and privileged mode. A High or increasing number of switches should be looked at.

System: System Up Time

This counter can be used as a measure of system availability.

Two additional counters that could prove useful for detailed thread analysis are:

Thread: Thread State

State=1 means the Thread is Ready and Waiting in the dispatching queue. If this occurs to excess for a thread, you need to investigate a potential CPU resource issue.

Thread: Thread Wait Reason

Can be used to investigate why threads that are waiting are not getting service. Can help to isolate components that are overloaded.

Monitoring Disk Usage

Note: I have slightly altered the naming convention for the disk section. The objects are given as P/L Disk. This indicates that the information is available in both the Physical and Logical Disk objects. Be particularly interested in the logical disk object/counters for the disk containing the paging file.

P/L Disk: Current Disk Queue Length

Instantaneous count of queued I/O requests. Systematically under sampled on a uniprocessor, but a good number to compare to the calculated value for the Avg. Disk Queue Length.

P/L Disk: Avg. Disk Queue Length

This is the calculated average number of active disk requests and is equal to Disk Transfers/sec * Avg. Disk sec/Transfer. If this value is consistently greater than 2-3 per disk and the Disk Transfers/sec is high (>100), the disk drive (or partition) is becoming a bottleneck. Also review:

P/L Disk: Avg. Disk Read Queue Length

P/L Disk: Avg. Disk Write Queue Length

P/L Disk: Disk Transfers/sec

This is the rate of operations (I/Os per second) for the selected disk during the sample interval. If this value rises above 100 for a single physical disk and the Avg. Disk sec/Transfer is higher than your baseline, it is the disk drive that is the bottleneck. Also look at:

P/L Disk: Disk Reads/sec

P/L Disk: Disk Writes/sec

P/L Disk: Avg. Disk sec/Transfer

Overall measured average response time of all disk requests. As a general rule, values above 0.035 seconds indicate slow response times at the device. Also look at:

P/L Disk: Avg. Disk sec/Read

P/L Disk: Avg. Disk sec/Write

P/L Disk: Disk Bytes/sec

This is the rate at which bytes are transferred to or from the disk. Sum this counter for all disks attached to the same SCSI/Fiber channel. If this value is 80% or higher than the rated throughput of the channel, the channel is the bottleneck. Also look at:

P/L Disk: Avg. Disk Bytes/Read

Calculated average block size = Disk Read Bytes/sec / Disk Reads/sec.

P/L Disk: Avg. Disk Bytes/Write

Calculated average block size = Disk Write Bytes/sec / Disk Writes/sec.

P/L Disk: Split I/O/sec

This is the rate that I/Os were split into Multiple I/Os. Splits result from requesting data in a size that is too large to fit into a single I/O or from a fragmented disk. If you see numbers higher than your baseline, run defrag.

P/L Disk: % Idle Time

New in Windows 2000.

% Disk busy is not accurate in NT releases.

This is the percentage of time that the disk is not in use. It can be used in the following computations to supply several valuable metrics:

Used to calculate: Disk utilization = 100 - % Idle Time.

Disk service time = Disk utilization / Disk Transfers/sec.

Disk Queue time = Avg. Disk sec/Transfer - Disk service time

Monitoring Network Usage

Network Interface: Bytes Total/sec

This is the rate at which all bytes are sent and received on the selected network interface, including those bytes used as overhead (framing, etc.). You should also look at:

Network Interface: Total bytes received/second

Network Interface: Total bytes sent/second

Network Interface: Output Queue Length

This is the length of the output packet queue (in packets). If this value is greater than 3 for sustained periods of time (more than 15 minutes), the selected interface is becoming a network bottleneck. This rule of thumb is valid only for single-CPU systems.

Network Segment: % Network Utilization

Percentage of bandwidth in use on this network segment. For Ethernet based network segments, a value consistently in the 50-70% range indicates a problem. Response times on the network may be elongated. Switched segments can run at close to 90% utilization without degradation.

Network Interface: Current Bandwidth

You can also calculate % Network utilization = (Total bytes received/second + Total bytes sent/second)/ Current Bandwidth.

Another approach to isolating a network bandwidth problem is to sum Server: Bytes Total/sec for all servers on the network. If this approaches the maximum transfer rate, you should consider segmenting.

Network Interface: Packets sent/sec

Rate of packets sent per second on the selected network interface.

Network Interface: Packets Outbound Errors and Received Errors

This is the count of outbound packets that could not be transmitted/received because of network errors. If this value is > 1, the selected network interface is experiencing problems that are causing the network to slow so that the system can handle the errors and retransmit the data. The problem could emanate from any NIC or network device connected to the segment.

Redirector: Current Commands

Counts the number of commands queued for network file processing. Should never be more than one per NIC. Higher numbers indicate a potential serious bottleneck.

Server: Work Item Shortages

This is the number of times that STATUS_DATA_NOT_ACCEPTED was returnd at receive indication time. It indicates that insufficient MaxWorkItems or InitWorkItems have been allocated. These can be tuned in the registry under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer. They should be tuned until they are = 0, unless Processor: % Processor Time is also > 80%.

Server: Pool NonPaged Failures &

Server: Pool Paged Failures

The number of times that allocations from NonPaged & Paged pools have failed. Network operations uses these memory pools. Values of > 1 on a regular basis indicates that there is insufficient physical memory on the server to support network operations.

For network clients using **NT Server's file and print sharing services**, you should review the following counters in the **Redirector** object:

Bytes Received/sec

Bytes Transmitted/sec

Read Bytes Network/sec

Write Bytes Network/sec

File Data Operations/ sec

For machines using **Microsoft's IIS Web server**, The following objects and counters should be collected and analyzed:

Internet Information Services Global: File Cache Hits %

Caching HTML file handles in memory saves on disk I/O and improves performance. Static html files are cached in system's file cache using the efficient MDL interface.

Internet Information Services Global: Current Files Cached

Internet Information Services Global: Current File Cache Memory Usage

Internet Information Services Global: BLOB Cache Hits %

Caching Binary Large Objects (BLOBs, GIFs and JPEGs associated with HTML requests) in memory saves on disk I/O and improves performance.

Internet Information Services Global: Current BLOBs Cached

Web Service: Total Method Requests/sec

The HTTP Method defines the type of HTTP request. This is the total Web service request rate per Web site.

Web Service: Get Requests/sec

GET is the HTTP Method used to request static HTML pages and embedded GIF and JPEG requests.

Web Service: CGI Requests/sec

CGI scripts are involved in many forms processing applications.

Web Service: ISAPI Extension Requests/sec.

ISAPI requests predate Microsoft's Active Server Pages server host scripting facility to support dynamic html pages.

Web Service: Current Connections

Web Service: Current CGI Requests

Web Service: Current ISAPI Extension Requests

Web Service: Service Uptime

Uptime for each Web site instance for tracking availability.

Active Server Pages: Request Execution Time

Execution time for the last ASP request (in milliseconds).

Active Server Pages: Request Queue Time

Queue time delay for the last ASP request (in milliseconds).

Active Server Pages: Requests Executing

Active Server Pages: Requests Queued

Active Server Pages: Requests/sec

The analyst should calculate average response time = (Requests Executing + Requests Queued) / Requests/se**c** (from Little's Law). Compare this value to the reported Request Execution Time + Request Queue Time, which reports the execution and queue time of the last completed ASP request only.

Active Server Pages: Transactions/sec

Active Server Pages: Transactions/pending

Monitoring Memory Usage

Memory: Available Bytes

This is the amount of physical memory available to processes running on the system. Process Working Set growth becomes constrained when Available Bytes < 4 MB (approximately), particularly if the disk where the pagefile is located is busy; the system is under severe stress when Available Bytes < 1 MB.

Memory: Pages Output/sec

This is the number of pages written to the disk pagefile to free up space in physical memory. Pages are written to disk only if changed in memory, so they probably hold data, not code. Try to limit Pages Input/sec + Pages Output/sec to 10-20% of total disk bandwidth, if possible. Disk bandwidth absorbed for paging operations is unavailable for application processes. If you have a combination of high Pages Output/sec, low Available Memory, and a busy pagefile disk, you are running short of RAM.

Logical Disk: Transfers/sec (for the pagefile.sys drive)

This is the metric that will measure a busy pagefile disk. See the comments in the previous counter discussion.

Memory: Cache Bytes

Actually, the System address space working set, but includes the file cache. The sum of Pool Paged Resident Bytes + System Code Resident Bytes + System Driver Resident Bytes + System Cache Resident Bytes. This counter gives you an idea of how much physical RAM is being used for dynamic system cache. Here is a brief discussion of the components:

Memory: Pool Paged Resident Bytes

The Operating system's pageable memory that is currently resident in RAM.

Memory: System Code Resident Bytes

Memory utilized by system code.

Memory: System Driver Resident Bytes

Memory utilized by driver routines that are resident in memory.

Memory: System Cache Resident Bytes

The current amount of RAM used for the file cache.

Paging File: % Usage

The percentage of the paging file instance in use. If this is consistently a high number, consider adding RAM to your system. A yardstick is to add RAM equal to the size of pagefile * % pagefile in use.

Server: Pool NonPaged Failures &

Server: Pool Paged Failures

The number of times that allocations from NonPaged & Paged pools have failed. Network operations uses these memory pools. Values of > 1 on a regular basis indicates that there is insufficient physical memory on the server to support network operations.

Server: Pool NonPaged Peak:

This counter can be used as an estimate of how much memory you actually need.

Memory: Committed Bytes

Represents virtual memory pages backed in either RAM or secondary storage (paging files). Calculate a virtual:real memory contention index = Committed Bytes / Total RAM. Consider adding RAM when this ratio starts to approach 2:1.

Memory: Commit Limit

Maximum number of virtual memory pages that can be allocated without extending the paging file(s).

Memory: % Committed Bytes in Use

Committed Bytes / Commit Limit. Consider adding RAM when consistently > 70% on a Server. On a Workstation, start closing some applications.

Memory: Page Faults/sec

Can be a grossly misleading number. Page Faults/sec = "soft" Transition Faults + application file Cache Faults + demand zero faults + hard page faults.

Memory: Page Reads/sec

Hard page fault rate! This is the one to really watch. Hard faults are I/Os. Adding RAM will convert it back to a soft fault, which is more bearable.

Memory: Pages Input/sec

You can calculate the bulk paging rate: Pages Input/sec / Page Reads/sec.

Memory: Pool Paged Bytes

Develop a virtual:real memory contention index = Pool Paged Resident Bytes / Pool Paged Bytes. Compare to Page Reads/sec in order to anticipate real memory bottlenecks.

Memory: Pool Nonpaged Bytes

The System's nonpageable (fixed) memory.

Memory: Page Writes/sec

Updated "dirty" pages must be flushed to disk before they can be reused by a different application.

Memory: Transition Faults/sec

"Soft" page faults resolved without having to access the disk. Interesting number to track, but there is nothing you can do about the Transition Fault rate. Consider them a by-product of the Windows 2000 page stealing algorithm.

Memory: Cache Faults/sec

Normal application file I/O operations are diverted to use the paging subsystem.

Memory: Demand Zero Faults/sec

The rate at which applications require brand new pages.

Memory: Write Copies/sec

Private Copy on Write pages from shared DLLs.

Note: The analyst can develop a measure of the amount of time that the system is spending on paging by: Memory: Pages/sec * Logical Disk: Average Disk sec/Transfer (for the pagefile disk). Convert the resulting seconds into a percentage of time in the measured interval. The value should not exceed 10%. Greater than 20% indicates a thrashing situation.

Other Counters of Interest

In the **Process** object, the following counters may be of interest:

% Processor Time

Per Process CPU usage.

Page Faults/sec

Indirect indicator of per Process I/O activity in NT 4.0; but includes "soft" transition fault activity.

IO Data Bytes/sec

New in Windows 2000. Tracks logical requests, so do not expect to match these new per Process IO Counters with disk activity measurements.

IO Data Operations/sec

New in Windows 2000.

Virtual Bytes

Per Process contribution to Committed Bytes. Cautionary note: shared DLL virtual bytes are not counted as part of the Virtual Bytes of each associated Process.

Working Set

Pages from resident shared DLLs are counted as part of the Working Set of each associated Process.

Elapsed Time

Used to track application availability.

For **file and print servers**, review:

In the **Server** object:

Bytes Received/sec

Bytes Transmitted/sec

Server Sessions

Context Blocks Queued/sec

The rate of individual file server requests sent by networking clients

And in the **Print Queue** object:

Jobs

Helpful new performance object in Windows 2000. Current number of jobs in a print queue.

Bytes Printed/sec

Total Jobs Printed

Total Pages Printed

Corrective Measures

As you can see, even this “short list” yields a daunting task of data collection and analysis. Building a baseline understanding of system performance will allow for rapid detection of resource constraints, isolation of troublesome components and prediction of impending bottlenecks before they develop into serious situations. The topic of tuning NT is large enough to deserve, at least, equal time as the measurement issue. Clearly, we cannot address it here. We will, however, whet your interest with some

quick lists of tuning options available for each of the resources discussed here.

Tuning strategies

Memory

- Select the appropriate NT Memory strategy from Control Panel/Network/Services/Server/Properties.
- Optimize virtual memory and the paging file system
- Remove unnecessary processes from the server
- Schedule memory intensive jobs to off hours.
- Last resort – Add RAM

Disk

- Distribute file system activity
- One logical disk per physical disk
- Group similar disk work.
- Use RAID appropriately.
- Last resort – Add hardware.

Network

- Balance network loads
- NIC settings
- Device drivers and BIOS levels
- Last resort – Faster NICs

CPU

- Make sure that the root cause is not within another resource area.
- Remove CPU overhead
    - Wasteful hardware components
    - Don't implement compression
    - Offload CPU intensive work.
- Remove faulty Hardware.
- NT Service Packs
- Last resort – Upgrade the CPU.

Closing Advice

Monitoring any system that is running mission critical work is an undisputed requirement of sane IT management. With the wealth of data in NT, you cannot watch everything all of the time with the staff that you are likely to get assigned to the task. Therefore, the following guidelines are critical to a successful program:

- Limit the metrics that you collect and analyze.
- Build an historical database of performance metrics for the key systems in your environment.
- Develop reports and graphs that allow for a quick high-level review of system performance.
- Conduct this high level review on a regular basis.
- Through this review process, develop an understanding of the baseline values for normal, acceptable behavior of these systems.
- Be on the lookout for any changes in this expected behavior pattern.
- Use the metrics to isolate the resource that is the bottleneck (or potential bottleneck).
- Take corrective action.
- Go back and make sure that the correction was effective.
- Start regular monitoring again.