



## **Performance Counters for SQL Server: What to Watch and What They Mean**

**Summary:** Provides testers, developers and database administrator with information on Microsoft SQL Server 2000 database performance counters describing the when and why to watch these counters.

© 2003 Microsoft Corporation. All rights reserved.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft is a registered trademark of Microsoft in the United States and/or other countries.

# Table of Contents

<b>Performance Monitoring .....</b>	<b>4</b>
Not 'all' is good.....	4
Monitoring time-periods.....	4
Data points .....	5
Main Performance Counters .....	5
<b>1. Memory:</b> .....	<b>5</b>
<b>2. Network Segment (Interface):</b> .....	<b>5</b>
<b>3. Disk I/O</b> .....	<b>6</b>
<b>4. Processor</b> .....	<b>6</b>
<b>5. System</b> .....	<b>7</b>
<b>6. SQL Server counters</b> .....	<b>8</b>

# Performance Monitoring

Databases by their inherent nature are always changing. It could be that just the data, which is changing or the requirement for data access (data query) changing. Therefore, it is not only important, but also critical that the database performs optimally. A good database performance is essential to virtually any application. Hence, database performance monitoring is a constant and continuous process.

A few performance counters act as the main indicators for the performance of the database server system. Then to further drill-down into the bottlenecks, we monitor other counters that are listed below in this paper.

## Not 'all' is good

Note that monitoring all the counters all the time is not recommended because it typically skews the log data for counters due to extra resources required to log all the counters ([Schroedinger's cat](#)). Therefore, always start with a very limited, high level of monitoring and drill down or switch on other relevant counters as and when required. Also, do not forget to turn off the monitors that are not required.

It is also important to group the data based on similar scales. Some counters are reported in percentages whereas some are on an absolute scale from 1 to 1000 or .001 to 1. This grouping will help in getting consistent comparisons and interpreting clear trends between the counters.

## Monitoring time-periods

Typically, based on the various uses of the performance data logs, the time-periods can be classified into three periods:

- 1. Daily:** This type of data is very useful in performance monitoring and troubleshooting because this is granular enough to give us an idea about what areas to explore to pinpoint any problems. Depending upon what problems are identified, it is easy to analyze just a few hours of data to get an idea of the loads on the system, memory state, CPU processing time, etc.
- 2. Monthly:** This type of data is typically useful in database maintenance issues for example index tuning, re-building the indexes or creating back-up plans. This data is not granular enough for troubleshooting and other performance optimizing diagnosis.
- 3. Quarterly:** This type of data is typically useful for trend analysis. The data at a quarterly level will help in predicting future requirements based on growth of the database and requests to the database. This is also a good indicator for preparing for high/low demand seasons for example, a database serving an online-retail store during holidays.

## Data points

As a general rule of thumb, a 3-second counter delay is typical in tests that run 0.5 hours to 4 hours. Longer tests should adjust sampling intervals to 10-15 seconds. Shorter tests might require 1-second intervals.

Another way of looking at this would be to gather the data that have enough detail for the daily analysis. For monthly and quarterly analysis, you may want to aggregate the data daily data.

## Main Performance Counters

Following is a list of high-level performance counters typically monitored on a continuous basis.

1. **Memory:** By default SQL Server manages its memory requirement dynamically. When SQL Server needs more memory, it allocates the required memory from the OS. Of course, this is subject to the amount of memory available on the system. For optimized use of resources it is recommended to monitor the following counters for memory:

- a. **Available Bytes:** This is the snapshot of the free memory available for the system. This will need to be averaged over a period (say 24 hours) to get the trend. It is recommended that this counter be over 5000 KB. Low values indicate an overall shortage of physical memory.

- b. **Page Fault counter**

- i. **Page Faults/sec**

- ii. **Pages/sec**

These counters indicate the number of page faults for data and code pages and give us an indication of the number of times disk I/O and/or memory outside of SQL Server allocated range is accessed. These counters should be as low as possible. '0' is ideal and '>5' is a problematic number for **Pages/sec** counter.

Pages/sec specifically gives us the counter for data pages also known as hard pages. Page Faults/sec gives us both code and data or simply hard and soft pages. Hence to calculate the soft page faults/sec: **Pages Faults/sec - Pages Input/sec = Soft Page Fault/sec.**

2. **Network Segment (Interface):** This is an important area to monitor especially in situations when there is too much data activity over the network. It is a good indicator for validating the network card's throughput.
  - a. **Bytes Received/sec:** This counter gives you the number of bytes received by the system per second, averaged over the interval period

- b. Bytes Sent/sec:** The number of bytes sent by the system per second, averaged over the interval period
- c. Bytes Total/sec:** This is the total of number of bytes sent and received by the system, averaged over the interval period

- 3. Disk I/O:** Following are the main counters for measuring Disk I/O activity.
- a. % Disk Time (Physical & Logical):** The percentage of the time the disk was busy performing read or write functions. This % Disk Time value is the sum of the % Disk Read Time and % Disk Write Time counters. You use physical disk counters for single disk and logical disk counters if the volume spans multiple disks.  
As a general rule of thumb, this counter should not exceed more than **55%** for continuous periods. If this counter exceeds 55% for continuous periods, i.e. over 10 minutes during 24-hour period, then the SQL Server may be experiencing an I/O bottleneck. Some of the ways to increase disk I/O are:
    - i.** Adding drives to the array (RAID) and/or getting faster drives
    - ii.** Add more cache to the controller card and/or get a faster controller
    - iii.** Reconfiguring RAID for better performance. As a rule of thumb if there are **50-50 read/write operations or more writes then read then configure RAID 10**. If there are, **mostly or purely read operations then configure RAID 5**.
    - iv.** Defrag the disks

Actually, another interesting way of looking at these numbers would be to see how busy the disk is not. For example, if the %Idle Time is very low (say < 10%) for an extended period of time where as the %Disk Time is very high (say >100%), we have a disk bottleneck. Some of us have noticed this counter reaching up to 4500%!

- b. Avg. Disk Queue Length:** This counter is the actual disk queue for read and writes operations. A disk queue length of **2** is the maximum recommended value for this counter. Depending upon the scenario, the **physical** or **logical** disk queue length should be measured and taken into account. For example, if you have an array of 10 physical disks, and the Avg. Disk Queue Length is 16 for a particular array, then the actual Avg. Disk Queue Length for each drive is 1.6 ( $16/10=1.6$ ), which is well within the recommended 2 per physical disk. If you are measuring logical disk queue length then the whole array is considered as one drive and you get the disk queue length for that 'logical' drive. Therefore, in this case it would be 1.6.

- 4. Processor:** The primary purpose of monitoring processor counters is to ensure the CPU is not becoming the bottleneck. Since it is relatively not easy to add CPUs, it becomes critical to ensure that all the CPUs are equally handling the load on the multi-processor machines. In addition, another

recommendation is to select processors with a larger L2 cache.

**% Processor Time** is the main counter to be watched for performance monitoring of the processors. This counter measures the percentage of time the processor was busy during the observation period. A continuous load of over 80% utilization indicates that the machine is running at about maximum load, and a plan will need to be made for future growth. If the utilization is more than 80% (including 100%) for short durations then it is typically acceptable. A general rule of thumb is **10 minutes peak load in 24-hour period**.

It is also important to measure individual % Processor Time if the server is running on a multi-processor machine. Sometime one CPU is experiencing the peak loads while others are almost idle. In these cases, we need to explore options to better load balance the CPUs.

## 5. System: Following are the main system monitoring counters:

- a. **Processor Queue Length:** This counter measures the processor queue for all processors in a system. A value of more than 2 indicates the CPU as the bottleneck. To reduce this numbers you can add processors. You can also review the SQL Server database for optimizations such as index tuning which will reduce I/O and subsequently CPU processing time.

In some cases, it is possible that more number of indexes will increase disk I/O activity, but in these cases, we can compensate for increased disk I/O activity by configuring database so that we have data on one physical disk system and indexes on another.

- b. **Context Switches/sec:** This counter shows the combined rate at which all processors on the computer are switched from one thread to another. Context switches occur when a running thread voluntarily relinquishes the processor (well...actually OS does the pre-emptive task), is preempted by a higher priority, ready thread, or switches between user-mode and privileged (kernel) mode to use an Executive or subsystem service. It is the sum of the values of **Thread\Thread: Context Switches/sec** for each thread running on all processors on the computer and is measured in numbers of switches.

If this rate is greater than **5,000** context switches per CPU, we have resource contention problems. We can further analyze what actually is causing the contention problems by using tools such as WinDBG. Once knowing the exact problems, we can add/configure hardware or in some cases even add another server to scale the system. Typically, this counter becomes important to watch if you experience increased Processor Queue Length.

With respect to SQL Server there are two approaches for lowering the context switching:

- i. **Affinity Mask:** An activity (thread) in a process can migrate from processor to processor, with each migration reloading the processor cache. Under heavy system loads, specifying which processor should run a specific thread can improve performance by reducing the number of times the processor cache is reloaded.

Using the affinity mask option increases performance on symmetric multiprocessor (SMP) systems (with more than four microprocessors) operating under heavy load. You can associate a thread with a specific processor and specify which processors SQL Server will use. You can also exclude SQL Server activity from processors given specific workload assignments by the OS.

Excluding SQL Server threads from running on particular processors helps the system's handling of processes specific to the OS.

- ii. **Lightweight Pooling:** When using the lightweight pooling option, SQL Server switches to a fiber-based scheduling model rather than the default thread-based scheduling model. Fibers are essentially lightweight threads. Use the command `sp_configure 'lightweight pooling',1` to enable fiber-based scheduling or you can use Enterprise Manager to change it. Fibers are scheduled by SQL Server instead of OS so there is less CPU load.

**6. SQL Server counters:** There are a number of specialty counters for dedicated servers. Some of the more important counters for database servers are listed here:

- a. **Cache Hit Ratio (SQLServer: Cache Manager Object):** This is the percentage of time that a record was found in cache. The recommended cache hit ratio is **90 percent** or more. For typical OLTP systems, the recommended value is **99%**. This can be achieved by adding more RAM to the system and a value of 90% is considered a low number.  
In OLAP systems, this percentage can be much less due to the nature of how OLAP systems work. Hence, the recommended ratio of **90%**.
- b. **User Connections (SQLServer: General Statistics object):** This is the value of number of users connected to this database. Note that this value is the total number of user connections and not the total number of users. By default, SQL Server is configured for **255** user connections. If this value is exceeded, then increase the value of "**Maximum Worker Threads**" to a number higher than 255. As a rule, this value should be higher than the number of user connections. You can also set the **maximum concurrent user connections** to '0', which means that the number of connections is limited only by the SQL Server maximum.
- c. **Transactions/sec (SQLServer: Databases object):** The number of transactions started for the database. These transactions come in the form of requests from client machines that are serviced by the database. Having too many concurrent transactions can cause locking problems and incur additional overhead.

- d. Data File(s) Size (KB) (SQLServer: Databases object):** The cumulative size of the data files that reside on a disk array. This counter is useful in tracking disk usage growth for capacity planning studies.
  
- e. Percent Log Used (SQLServer: Databases object):** The percentage of the log that is used. This counter is helpful in tracking log growth in capacity planning studies.

## Finding More Information

- SQL Server Books Online provides information on SQL Server architecture and database tuning along with complete documentation on command syntax and administration. SQL Server Books Online can be installed from the SQL Server installation media on any SQL Server client or server computer.
- For the latest information on Microsoft SQL Server, including technical papers on SQL Server, visit the public Microsoft SQL Server Web sites at:  
<http://www.microsoft.com/sql>  
<http://www.microsoft.com/technet/sql>  
<http://msdn.microsoft.com/sqlserver>
- An external resource that provides good information in the form of a periodical can be found at <http://www.sqlmag.com>. You will find many optimization and tuning hints, code samples, and insightful articles outlining the internal workings of SQL Server and other valuable information.
- Another non-Microsoft SQL performance-tuning site is <http://www.sql-server-performance.com>. This site contains an abundance of categorized information on performance tuning.