Windows Performance Counters Explained

Table of Contents [show]

Introduction

One of the most underutilized features of the Microsoft Windows® Operating System is it's excellent Performance tracking tool called Perfmon (Performance Monitor). As we started working with Enterprise Backend Systems, we quickly identified a huge need for an easy way to pull data out of the Perfmon tool to allow AppAdmins and other IT Professionals to quickly identify system performance issues.

When we started developing our BLG Performance Report Generator and SPASS – System Performance and Sizing Site we knew the time we put into researching and identifying the best performance counters to track would allow our products to be utilized by the majority of Windows Administrators. Our initial goal was to identify only the performance counters that immediately corresponded to the hardware within backend servers. To do this, we contacted many "Industry Experts" that specialized in Windows Performance Optimization, including some individuals inside of the Redmond company itself.

This guide covers all of the Performance counters that we decided to focus on. As you read this you will notice that some counters that are normally discussed are absent, such as `% Disk Time` , although these counters are important, we avoided any counters that requires additional calculations to be of benefit.

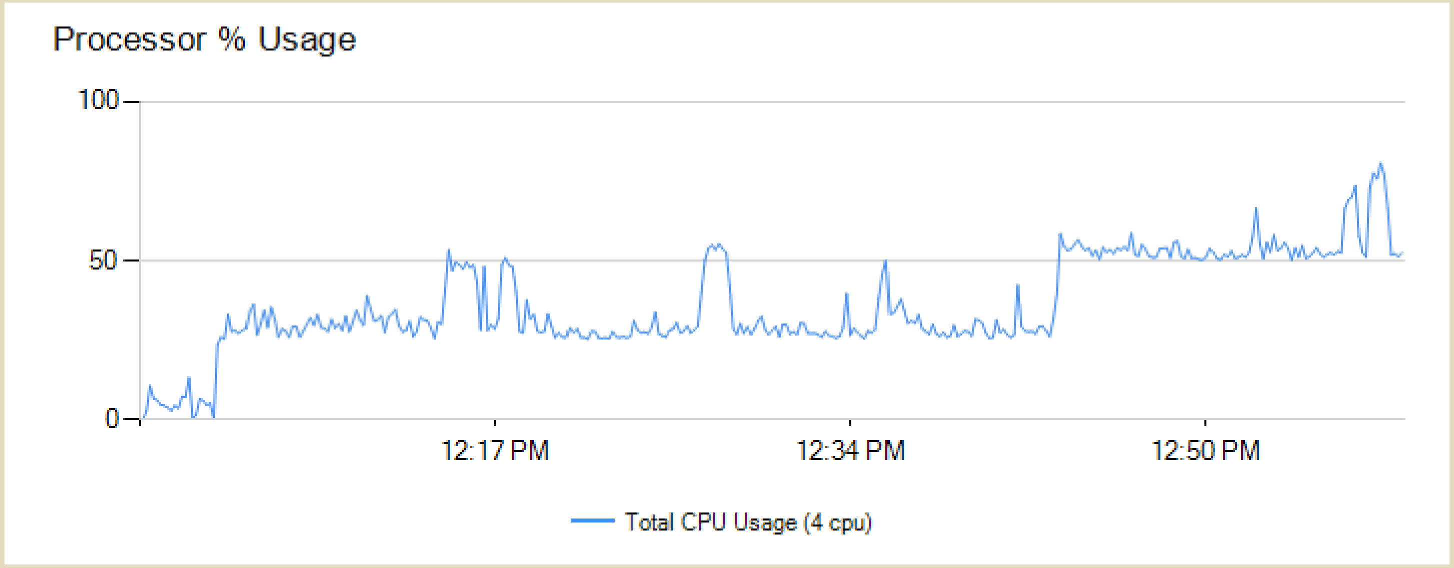
As you read through this, you will see sections similar to below. These include the description Microsoft provides within the perfmon tool for reference (*this *should* be legally allowed, but if Microsoft contacts us these may be removed in the future*).

Microsoft® Description – These boxes will contain the Counter information Microsoft Provides within Performance Monitor

Also note that all of the graphs were generated using our BLG Performance Report Generator and any reference toward adjusting settings to generate graphs are related to that application.

Processor

The Processor % Usage graph shows how much the processor(s) is being utilized. There are 2 graphs available for Processor Usage, one being the combined percent of all the processors on the system, the other being a graph of all the individual processors' usage. Normally, only the combined usage percent is generated, to create the individual stats graph you must enable the "Create Individual CPU Stats" checkbox under the options menu.

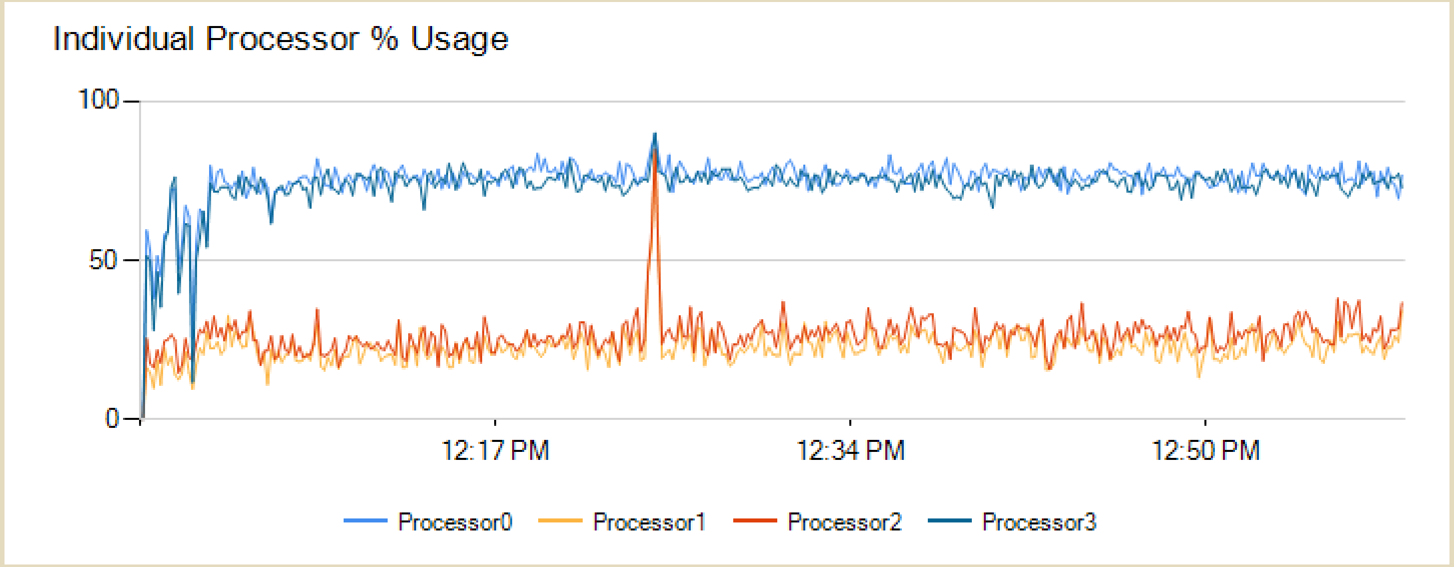


```
\Processor(*)\% Processor Time
```

On multi-core/CPU systems it is generally a good idea to first look at the combined CPU usage (especially when giving reports to non-technical people) since having 8, 16 or 32 graph points can be distracting and can cause some less than technical people to think there is a CPU issue if there are quite a few spikes on individual CPUs.

One thing to look for on the combined graph is to check for patterns. For instance, the above graph comes from a server running an ASP.net application that used quite a few third party dlls. As you can see, many of these dlls were single threaded and once the dlls were identified, they could be easily replaced or updated to use more than 1 CPU at a time. This would be extremely hard to view on a graph only showing the individual CPU stats since Windows® would move the process across multiple cores/cpus.

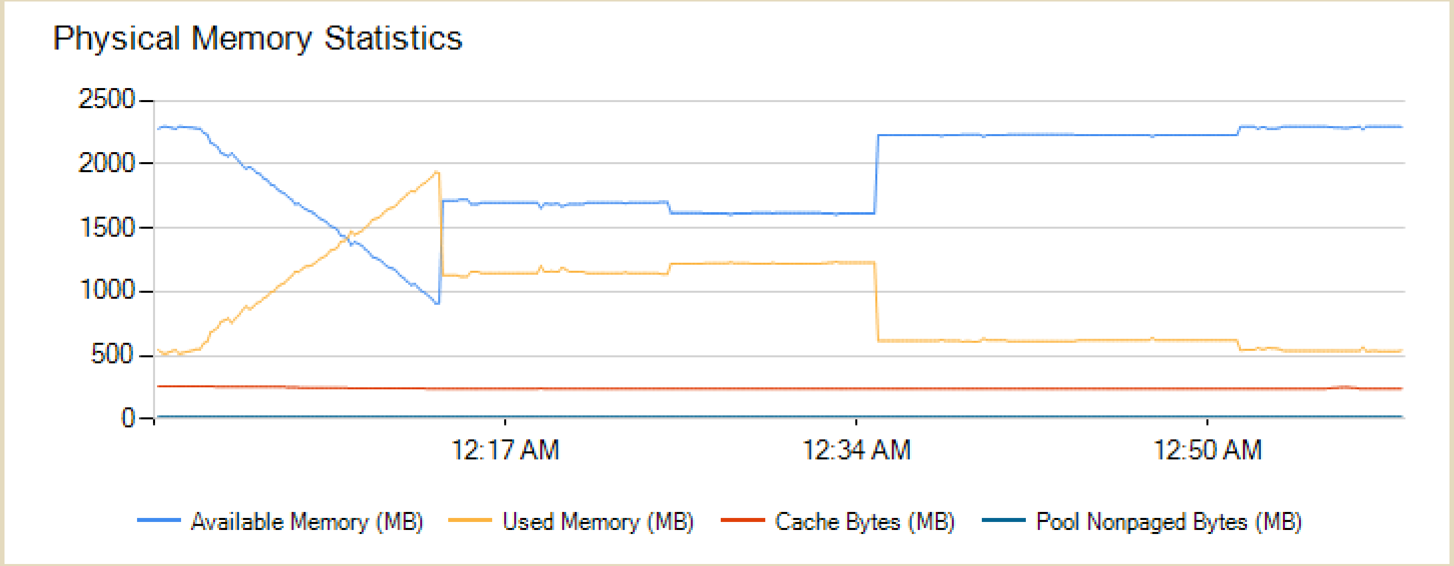
Microsoft® Description – % Processor Time is the percentage of elapsed time that the processor spends to execute a non-Idle thread. It is calculated by measuring the percentage of time that the processor spends executing the idle thread and then subtracting that value from 100%. (Each processor has an idle thread that consumes cycles when no other threads are ready to run). This counter is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval. It should be noted that the accounting calculation of whether the processor is idle is performed at an internal sampling interval of the system clock (10ms). On today's fast processors, % Processor Time can therefore underestimate the processor utilization as the processor may be spending a lot of time servicing threads between the system clock sampling interval. Workload based timer applications are one example of applications which are more likely to be measured inaccurately as timers are signaled just after the sample is taken.



Even though it is easier to see patterns in the combined CPU graph, there are times when it is recommended to also view the individual CPU stats, especially when a program is written to take full control of the processor(s). The above graph shows a program that has a data service that was written to only use 2 CPUs, even though the whole program was written to be load balanced across many servers, the data service was bound to 1 server. Since it was written this way, no matter how many application servers were added, the performance never increased since the bottleneck was the data service. Once the vendor updated their program to utilize more CPUs, the performance increased dramatically with an application “cluster”.

Physical Memory

The Physical Memory Statistics Graph shows the available Performance Counters for everything that consumes “Physical Memory” on a Windows® Server. In theory, adding all of these counters together should equal the amount of RAM installed, but in practice this does not work. This is the result of Windows ® using shared DLLs and some RAM allocations that are not counted anywhere (such as trimmed working set pages that have yet to be written to disk).



\Memory\Available Bytes

The Available Bytes counter is the pool size of available pages in RAM that the system uses to satisfy requests for new pages. There are multiple counters available for convenience that provide this information, such as Available Mbytes, but this program uses the Available Bytes counter and does the calculation when generating the graphs.

Usually when this value is approximately at least 10% of total system memory, the system has an adequate amount of memory. However when this value is below 10% you must use additional counters, such as Pages/sec to determine if System memory is adequate for the workload.

Note that the “\Memory\Available Bytes” counter is an instantaneous counter (sampled once during the measurement period).

Microsoft® Description – Available Bytes is the amount of physical memory, in bytes, immediately available for allocation to a process or for system use. It is equal to the sum of memory assigned to the standby (cached), free and zero page lists.

\Process(_Total)\Working Set

The Working Set counter is the total amount of resident pages allocated in RAM that all the processes that are running on the system can address without causing a Page Fault. The value of this counter is in bytes, but this program does the calculation to MB when it creates the graphs.

Note that the “\Process(_Total)\Working Set” counter is an instantaneous counter (sampled once during the measurement period).

Microsoft® Description – Working Set is the current size, in bytes, of the Working Set of this process. The Working Set is the set of memory pages touched recently by the threads in the process. If free memory in the computer is above a threshold, pages are left in the Working Set of a process even if they are not in use. When free memory falls below a threshold, pages are trimmed from Working Sets. If they are needed they will then be soft-faulted back into the Working Set before leaving main memory.

\Memory\Cache Bytes

The Memory pages that the System uses are counted in two main counters, Cache Bytes and Pool Nonpaged Bytes.

The Cache Bytes counter is the amount of resident pages allocated in RAM that the Kernel threads can address without causing a Page Fault. This counter includes the Pool Paged Resident Bytes, the System Cache Resident

Bytes, the System Code Resident Bytes and the System Driver Resident Bytes.

Note that the “\Memory\Cache Bytes” counter is an instantaneous counter (sampled once during the measurement period).

Microsoft® Description – Cache Bytes the size, in bytes, of the portion of the system file cache which is currently resident and active in physical memory. The Cache Bytes and Memory\\System Cache Resident Bytes counters are equivalent. This counter displays the last observed value only; it is not an average.

\Memory\Pool Nonpaged Bytes

The Pool Nonpaged Bytes counter is the amount of resident pages in RAM that the Kernel is using that cannot be paged out.

The Pool Nonpaged Bytes counter is also referred to as “Kernel Memory”. If you ever run into any issues with Kernel Memory note that it is extremely difficult to troubleshoot as you have to use Kernel level tools such as “poolmon” to track what is causing the issues.

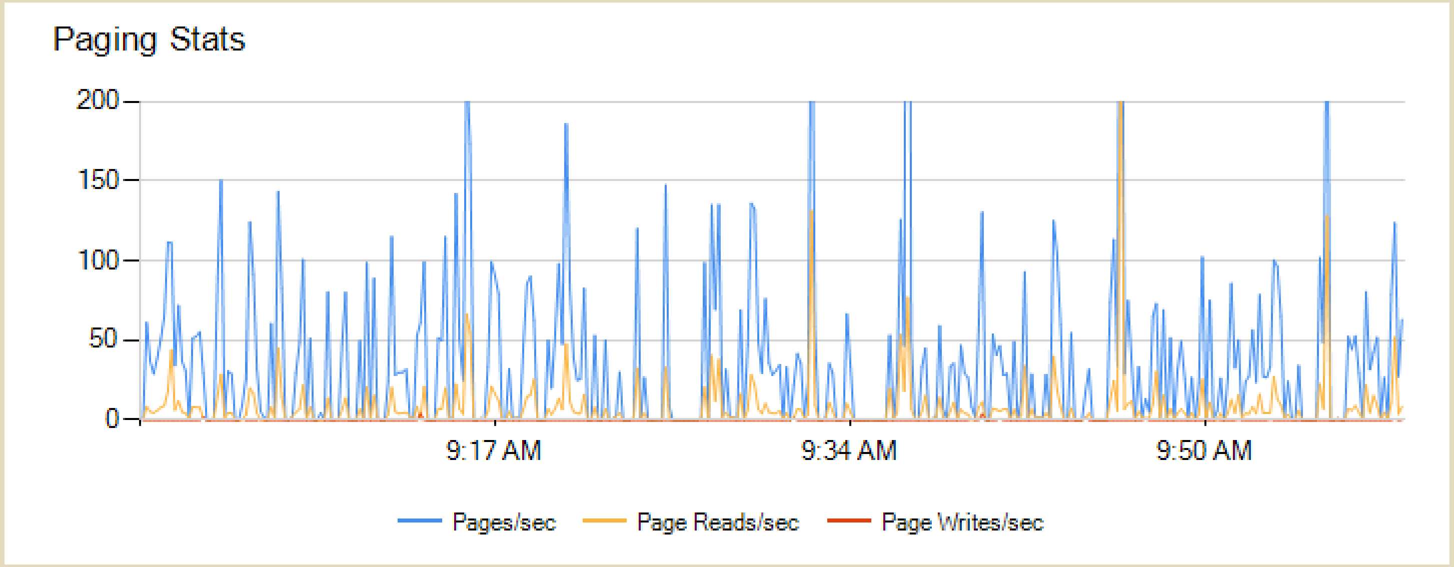
Also note that Microsoft Windows has limits on how much physical memory can be consumed by Kernel Memory. For instance Windows Server 2008 has a maximum limit of 75% of the amount of Physical Memory. If Kernel Memory reaches that Windows will in fact crash! Refer to the excellent WinInternals reference manuals for more information.

Note that the “\Memory\Pool Nonpaged Bytes” counter is an instantaneous counter (sampled once during the measurement period).

Microsoft® Description – Pool Nonpaged Bytes is the size, in bytes, of the nonpaged pool, an area of the system virtual memory that is used for objects that cannot be written to disk, but must remain in physical memory as long as they are allocated. Memory\\Pool Nonpaged Bytes is calculated differently than Process\\Pool Nonpaged Bytes, so it might not equal Process(_Total)\\Pool Nonpaged Bytes. This counter displays the last observed value only; it is not an average.

Paging

Memory shortages on Windows® can be identified by using the Paging counters available in Performance Monitor. The key counters that this program graphs are described below. When dealing with Windows ® Paging you have to keep in mind that paging occurs for various operations within Windows ® and excessive paging doesn't automatically indicate a memory shortage. For instance, many network card drivers utilize the Pagefile (sometimes excessively) and this can be misread as a memory shortage.



\Memory\Pages/sec

The Pages/sec counter is a primary indicator to determine if the system's memory is a bottleneck. The Pages/sec counter is the sum of both the Pages Input/sec and Pages Output/sec, some performance guides reference these directly, but we have decided to just graph the combined value.

A System with a sustained value over 20 should be closely monitored and a System with a sustained value of over 50 is probably lacking in System Memory. Again, it is normal for this to spike occasionally, especially if the other Memory counters do not show a lack of System Memory.

Microsoft® Description – Pages/sec is the rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the kinds of faults that cause system-wide delays. It is the sum of Memory\\Pages Input/sec and Memory\\Pages Output/sec. It is counted in numbers of pages, so it can be compared to other counts of pages, such as Memory\\Page Faults/sec, without conversion. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) non-cached mapped memory files.

\Memory\Page Reads/sec

The Page Reads/sec is another counter to watch for memory issues. This counter is a complement to the Pages/sec counter since it is directly related to how much the disk is affected from paging operations.

Microsoft® Description – Page Reads/sec is the rate at which the disk was read to resolve hard page faults. It shows the number of reads 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 working set or elsewhere in physical memory, and must be retrieved from disk. 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.

\Memory\Page Writes/sec

The Page Writes/sec is another counter to watch for memory issues. This counter is a complement to the Pages/sec counter since it is directly related to how much the disk is affected from paging operations.

Microsoft® Description – 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.

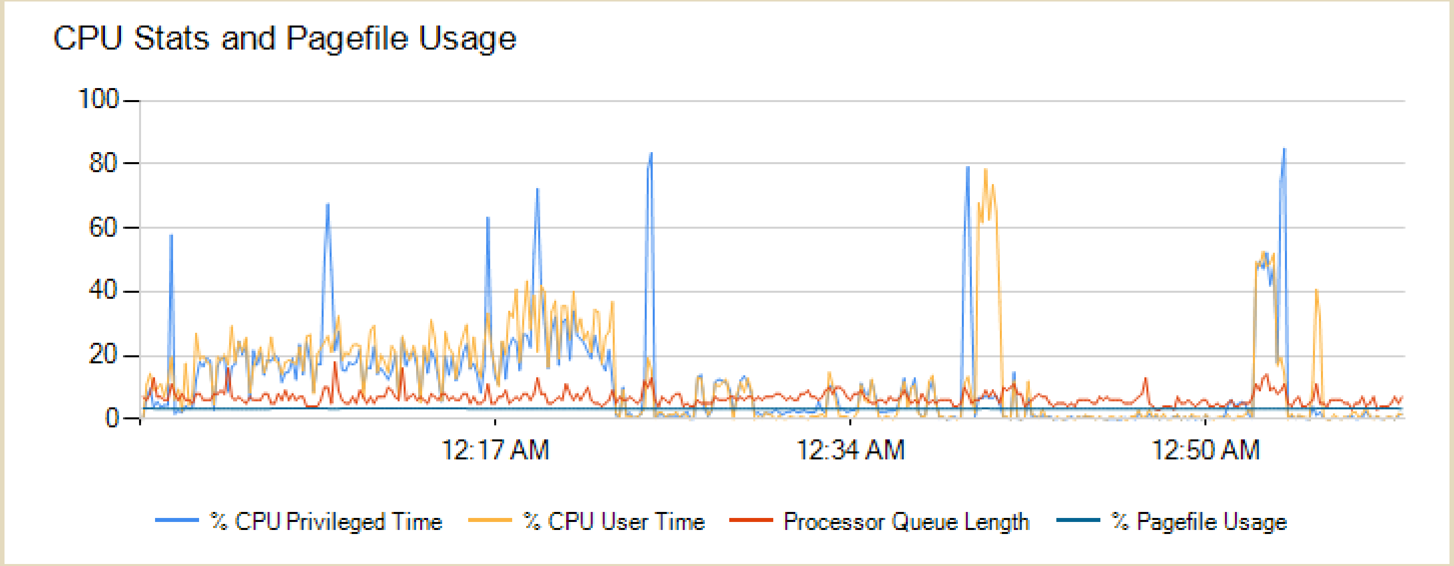
\Paging File(_total)\% Usage

The PageFile Percent Usage counter (shown on the next graph) shows what percent of the current pagefile is being used. Although not truly related to Performance, it is highly recommended to keep an eye on the Pagefile % Usage counter to ensure it does not reach 100%, which could cause severe system and application issues.

Microsoft® Description – The amount of the Page File instance in use in percent. See also Process\\Page File Bytes.

CPU/System Stats

In addition to the % Processor Usage counters above, there are additional counters that can detect how the Processor is being worked and if the Processor Subsystem could potentially be a bottleneck.



\Processor(_total)\% Privileged Time

The Processor(_total)\% Privileged Time counter shows the percent of time that the processor is spent executing in Kernel (or Privileged) mode. Privileged mode includes services interrupts inside Interrupt Service Routines (ISRs), executing Deferred Procedure Calls (DPCs), Device Driver calls and other kernel-mode functions of the Windows® Operating System.

Most of the time a processor should be executing User mode operations, a high % privileged time might indicate a poorly written device driver or a faulty piece of hardware.

Microsoft® Description – % Privileged Time is the percentage of elapsed time that the process threads spent executing code in privileged mode. When a Windows system service in called, the service will often run in privileged mode to gain access to system-private data. Such data is protected from access by threads executing in user mode. Calls to the system can be explicit or implicit, such as page faults or interrupts. Unlike some early operating systems, Windows uses process boundaries for subsystem protection in addition to the traditional protection of user and privileged modes. Some work done by Windows on behalf of the application might appear in other subsystem processes in addition to the privileged time in the process.

\Processor(_total)\% User Time

The Processor(_total)\% User Time counter shows the percent of time that the processor(s) is spent executing in User mode.

Microsoft® Description – % User Time is the percentage of elapsed time the processor spends in the user mode. User mode is a restricted processing mode designed for applications, environment subsystems, and integral subsystems. The alternative, privileged mode, is designed for operating system components and allows direct access to hardware and all memory. The operating system switches application threads to privileged mode to access operating system services. This counter displays the average busy time as a percentage of the sample time.

\System\Processor Queue Length

The Processor Queue Length shows the number of threads that are observed as delayed in the processor Ready Queue and are waiting to be executed.

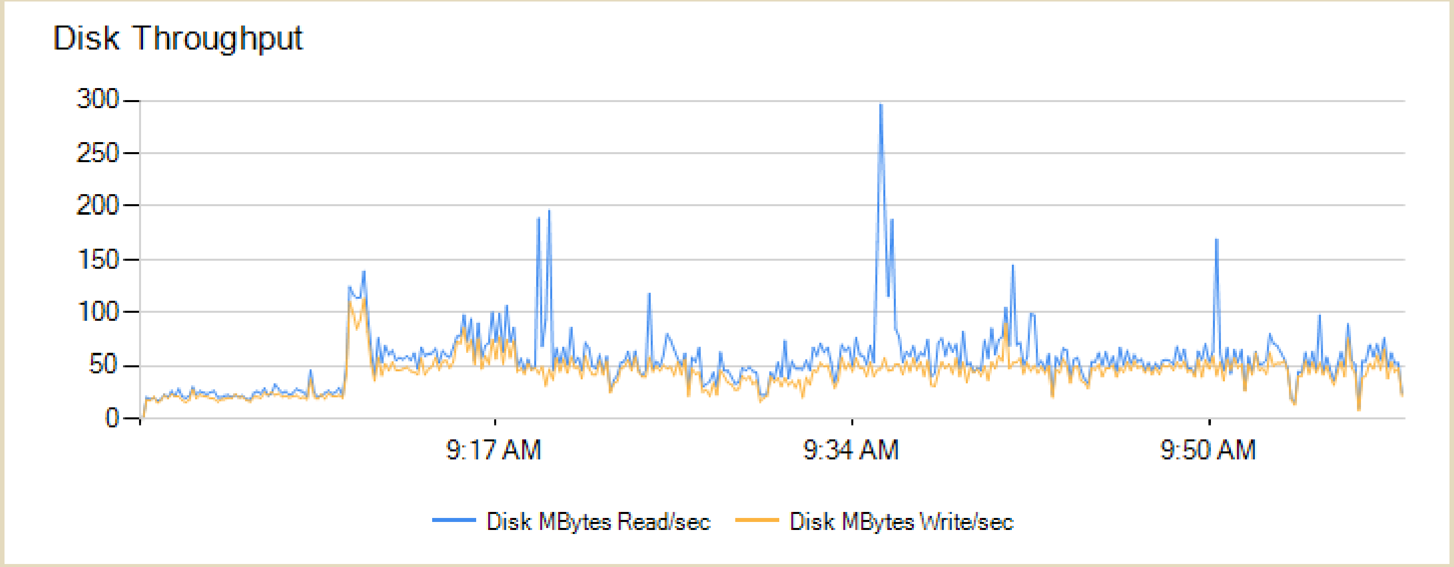
On single core machines, a sustained value over 5-10 or so may indicate that the workload on the system is more than the processor can handle. On multi-core systems, a good rule of thumb is to divide the Queue Length by the number of cores in the system and use the same indication above with that calculated value.

Note that the Processor Queue Length is an instantaneous counter, whereas the % Processor Usage counters are continuously measured counters that are averaged. There could be some disconnect between these counters because of this.

Microsoft® Description – Processor Queue Length is the number of threads in the processor queue. Unlike the disk counters, this counter counters, this counter shows ready threads only, not threads that are running. There is a single queue for processor time even on computers with multiple processors. Therefore, if a computer has multiple processors, you need to divide this value by the number of processors servicing the workload. A sustained processor queue of less than 10 threads per processor is normally acceptable, dependent of the workload.

Disk Throughput

The Disk subsystem could be a major system performance bottleneck, because of this we decided to focus on many disk performance counters and present the information by either graphing the total combined counters from all the Disks in the System or to graph the individual disk counters to allow you to pinpoint issues.



\PhysicalDisk(*)\Disk Read Bytes/sec

The Disk Read Bytes/sec shows the Disk Throughput of the Read Operation of the Disk. This is normally displayed in bytes, but we do the calculation when graphing the counter to display it in the standard MB/sec.

Microsoft® Description – Disk Read Bytes/sec is the rate at which bytes are transferred from the disk during read operations.

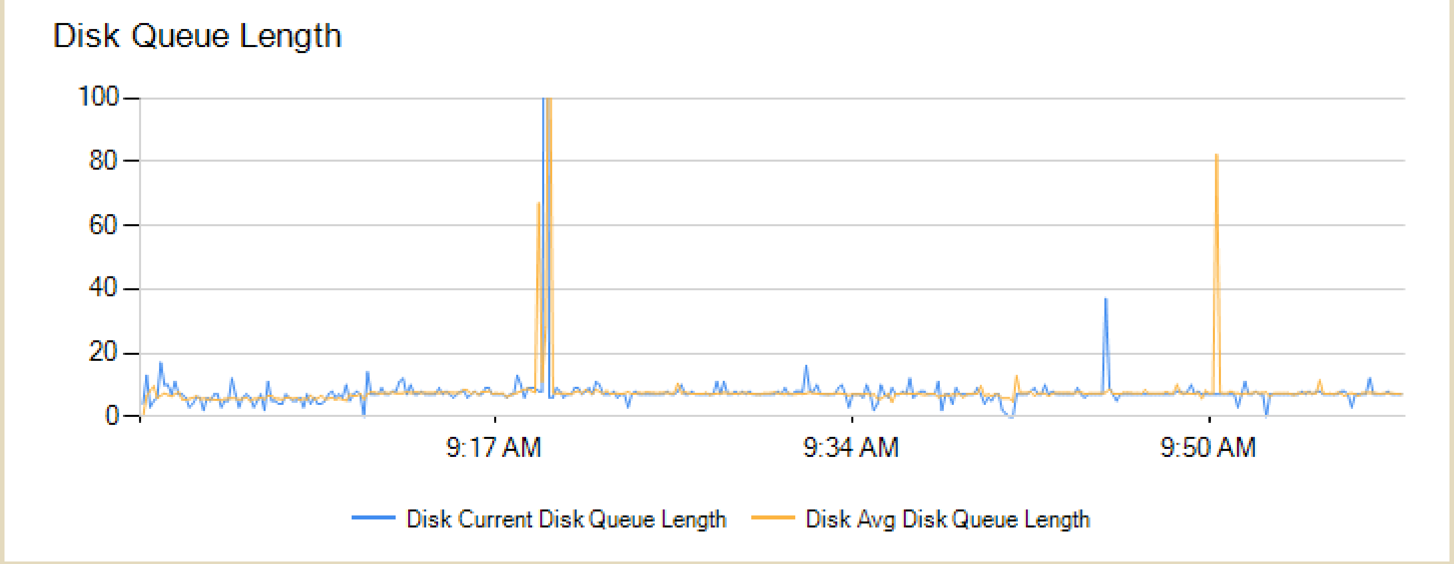
\PhysicalDisk(*)\Disk Write Bytes/sec

The Disk Write Bytes/sec shows the Disk Throughput of the Write Operation of the Disk. This is normally displayed in bytes, but we do the calculation when graphing the counter to display it in the standard MB/sec.

Microsoft® Description – Disk Write Bytes/sec is rate at which bytes are transferred to the disk during write operations.

Disk Queue Length

There are two main counters for the Disk Queue Length, the Average Disk Queue Length and the Current Disk Queue Length. There are arguments for only using either counter, so to stay out of the debate on which to use, we simply show values for both counters.



\PhysicalDisk(*)\Avg. Disk Queue Length

The Avg. Disk Queue Length counter is the “estimated” average number of requests that are either in process or waiting to be processed by the Disk.

Sustained counters above 5 per Disk could indicate a disk subsystem bottleneck. Remember to also keep an eye on the Pages/sec counter for low memory issues since low memory could cause Disk performance issues if the disk subsystem has to continuously process paging operations.

Microsoft® Description – Avg. Disk Queue Length is the average number of both read and write requests that were queued for the selected disk during the sample interval.

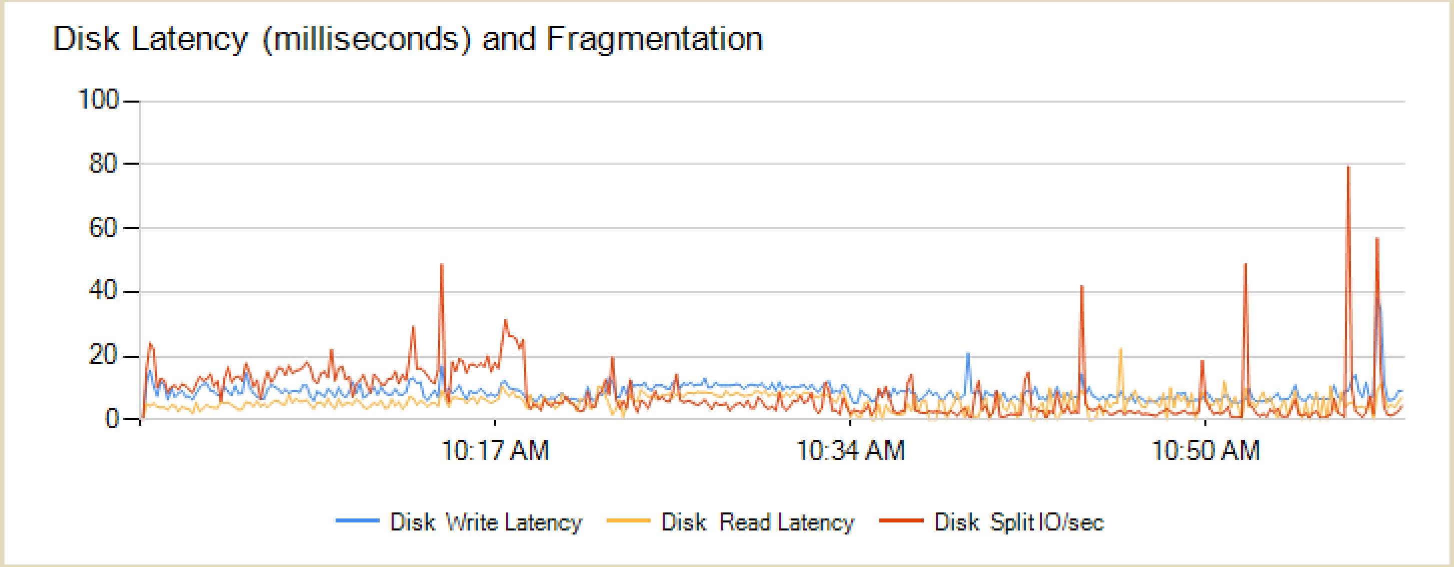
\PhysicalDisk(*)\Current Disk Queue Length

The Current Disk Queue Length counter is the actual number of requests that are either in process or waiting to be processed by the Disk at the time the counter is captured.

Sustained counters above 5 per Disk could indicate a disk subsystem bottleneck. Remember to also keep an eye on the Pages/sec counter for low memory issues since low memory could cause Disk performance issues if the disk subsystem has to continuously process paging operations.

Microsoft® Description – Current Disk Queue Length is the number of requests outstanding on the disk at the time the performance data is collected. It also includes requests in service at the time of the collection. This is a instantaneous snapshot, not an average over the time interval. Multi-spindle disk devices can have multiple requests that are active at one time, but other concurrent requests are awaiting service. This counter might reflect a transitory high or low queue length, but if there is a sustained load on the disk drive, it is likely that this will be consistently high. Requests experience delays proportional to the length of this queue minus the number of spindles on the disks. For good performance, this difference should average less than two.

Disk Latency and Fragmentation



\PhysicalDisk(*)\Avg. Disk sec/Read

The Avg. Disk sec/Read displays how long in milliseconds it takes for a read operation from the Disk. Watch this counter during times of heavy disk utilization to see if your system would benefit from a more robust Disk Subsystem (such as advanced RAID or newer flash drive technologies).

Microsoft® Description – Avg. Disk sec/Read is the average time, in seconds, of a read of data from the disk.

\PhysicalDisk(*)\Avg. Disk sec/Write

The Avg. Disk sec/Write displays how long in milliseconds it takes for a write operation to the Disk. Watch this counter during times of heavy disk utilization to see if your system would benefit from a more robust Disk Subsystem (such as advanced RAID or newer flash drive technologies).

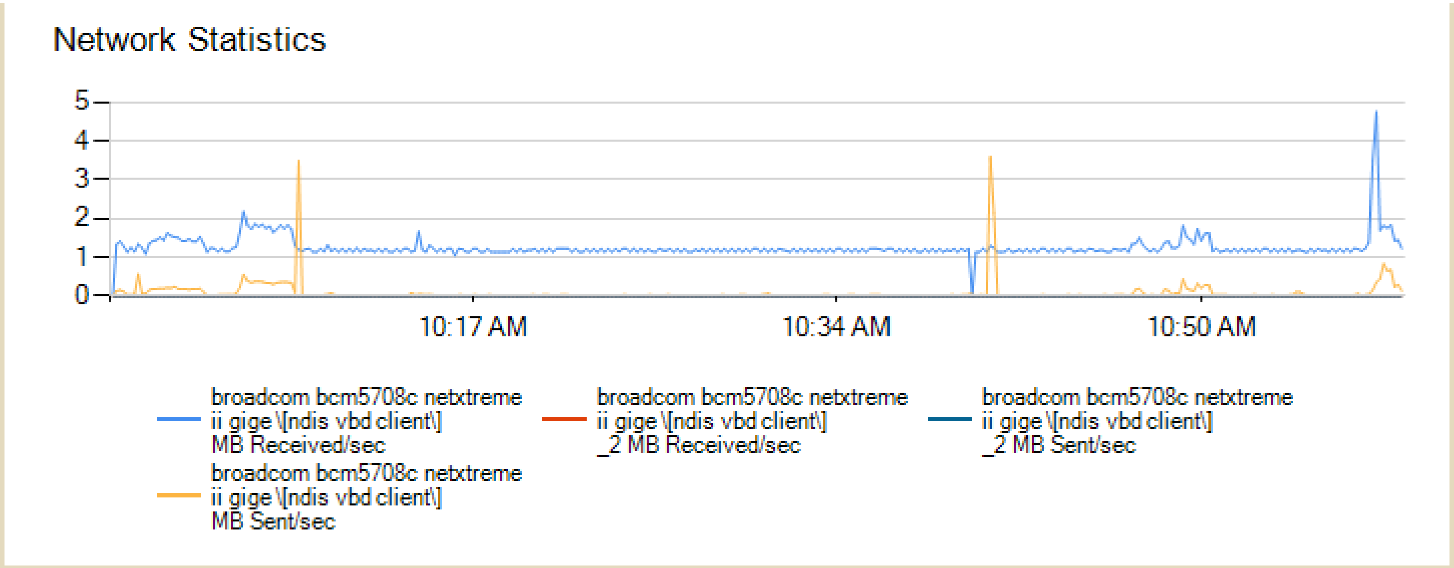
Microsoft® Description – Avg. Disk sec/Write is the average time, in seconds, of a write of data to the disk.

\PhysicalDisk(*)\Split IO/sec

The Split IO/sec counter displays the physical disk requests that are split into multiple requests. This counter is a primary indicator if a disk is fragmented and needs to be optimized.

Microsoft® Description – Split IO/Sec reports the rate at which I/Os to the disk were split into multiple I/Os. A split I/O may result from requesting data of a size that is too large to fit into a single I/O or that the disk is fragmented.

Network Throughput



\Network Interface(*)\Bytes Received/sec

The Network Interface Bytes Received/sec shows the Network Received Throughput of the NIC. This is normally displayed in bytes, but we do the calculation when graphing the counter to display it in the standard MB/sec.

Microsoft® Description – Bytes Received/sec is the rate at which bytes are received over each network adapter, including framing characters. Network Interface\Bytes Received/sec is a subset of Network Interface\Bytes Total/sec.

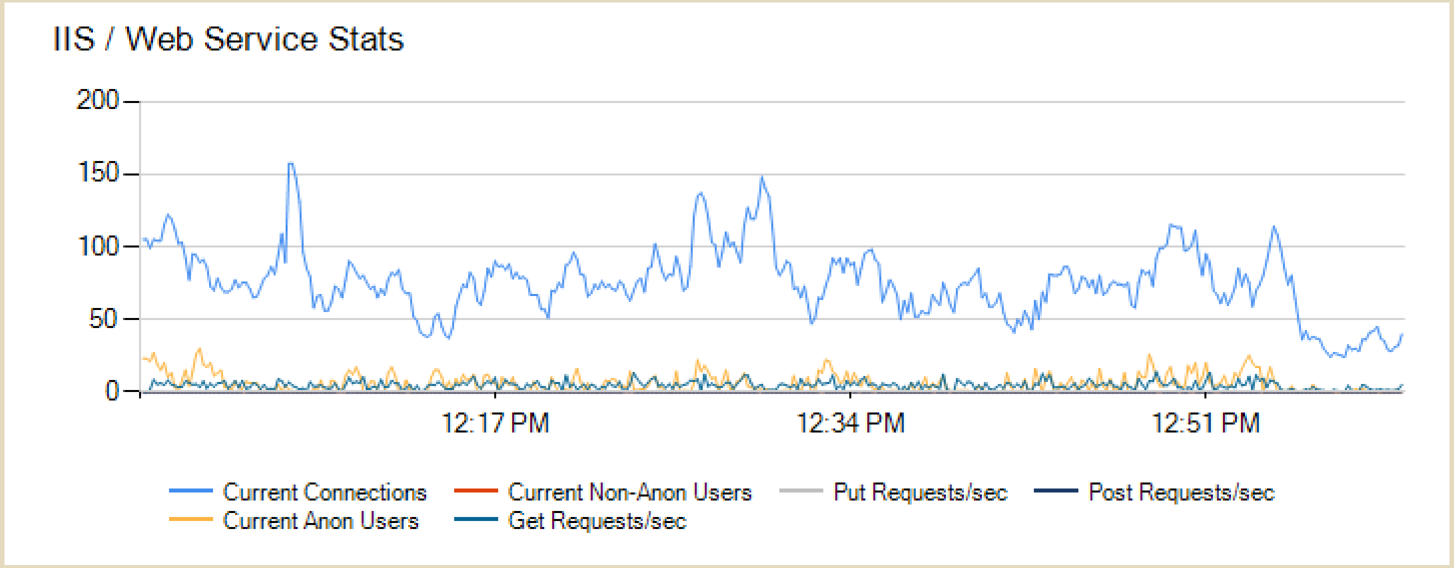
\Network Interface(*)\Bytes Sent/sec

The Network Interface Bytes Sent/sec shows the Network Send Throughput of the NIC. This is normally displayed in bytes, but we do the calculation when graphing the counter to display it in the standard MB/sec.

Microsoft® Description – Bytes Sent/sec is the rate at which bytes are sent over each network adapter, including framing characters. Network Interface\Bytes Sent/sec is a subset of Network Interface\Bytes Total/sec.

IIS / Webservice Counters

For graphing webservice stats, we focused on gathering the current user load of the services to allow you to monitor increased/decreased load over time and make the necessary adjustments. These counters don't really need additional explanation, so we just documented Microsoft® Description below.



\Web Service(_Total)\Current Connections

Microsoft® Description – Current Connections is the current number of connections established with the Web service.

\Web Service(_Total)\Current Anonymous Users

Microsoft® Description – Current Anonymous Users is the number of users who currently have an anonymous connection using the Web service.

\Web Service(_Total)\Current NonAnonymous Users

Microsoft® Description – Current NonAnonymous Users is the number of users who currently have a non-anonymous connection using the Web service.

\Web Service(_Total)\Get Requests/sec

Microsoft® Description – The rate HTTP requests using the GET method are made. Get requests are the most common HTTP request.

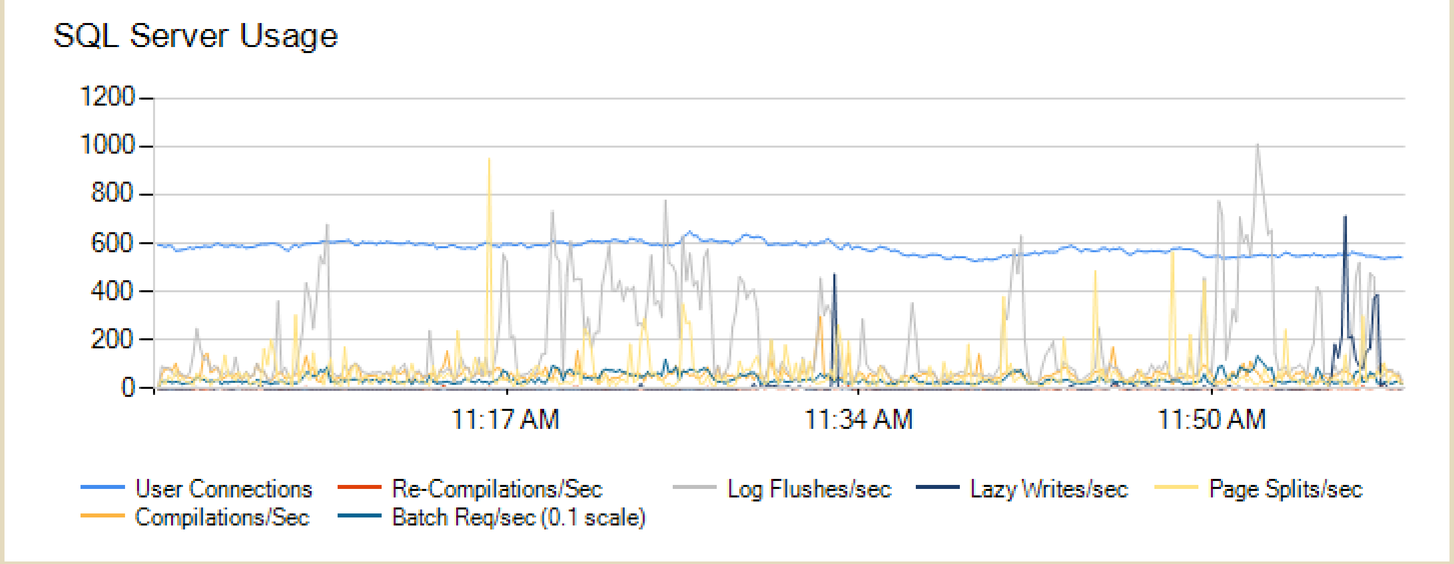
\Web Service(_Total)\Put Requests/sec

Microsoft® Description – The rate HTTP requests using the PUT method are made.

\Web Service(_Total)\Post Requests/sec

Microsoft® Description – The rate HTTP requests using the POST method are made.

SQL Server Usage Stats



\SQLServer:General Statistics\User Connections

The SQLServer User Connections counter allows you to monitor the number of connected users.

Microsoft® Description – Number of users connected to the system.

\SQLServer:SQL Statistics\SQL Compilations/sec

The SQL Compilations/Sec counter displays the number of times SQL Server compiles an execution plan per second. Note that compiling an execution plan is a resource-intensive operation.

Microsoft® Description – Number of SQL compilations.

\SQLServer:SQL Statistics\SQL Re-Compilations/sec

The SQL Re-Compilations/Sec counter displays the number of times SQL Server re-compiles an execution plan per second.

When the execution plan is invalidated due to some significant event, SQL Server will re-compile it. Re-compiles, like compiles, are expensive operations so you want to minimize the number of re-compiles.

Optimally, this counter should be less than 10% of the number of Compilations/Sec.

Microsoft® Description – Number of SQL re-compiles.

\SQLServer:SQL Statistics\Batch Requests/sec

Batch Requests/Sec counter shows the number of batches SQL Server is receiving per second.

This counter is a good indicator of how much activity is being processed by your SQL Server box.

Microsoft® Description – Number of SQL batch requests received by server.

\SQLServer:Databases(_total)\Log Flushes/sec

The SQLServer: Databases: Log Flushes/sec counter measures the number of log flushes per second. This graph shows the total Log Flushes across all Databases.

The log cache is a place in memory that SQL Server records data to be written to the log file. The log cache is used to roll back a transaction before it is committed, if the circumstances call for it. If this is a high value it may cause a disk bottleneck.

Every INSERT statement creates a Log Flush, if you are seeing a lot of Log Flushes it might be worthwhile to optimize your queries to reduce the amount of INSERT statements, which reduces Log Flushes, which will reduce Disk I/O

Microsoft® Description – Number of log flushes.

\SQLServer:Buffer Manager\Lazy Writes/sec

The SQLServer:Buffer Manage\Lazy Writes/Sec counter tracks how many time a second that the Lazy Writer process is moving dirty pages from the buffer to disk in order to free up buffer space.

Normally, this should not be a high value (less than 20/sec or so). Optimally, it should be close to zero. If this value is high, it might be an indication that more memory is needed.

Microsoft® Description – Number of buffers written by buffer manager's lazy writer.

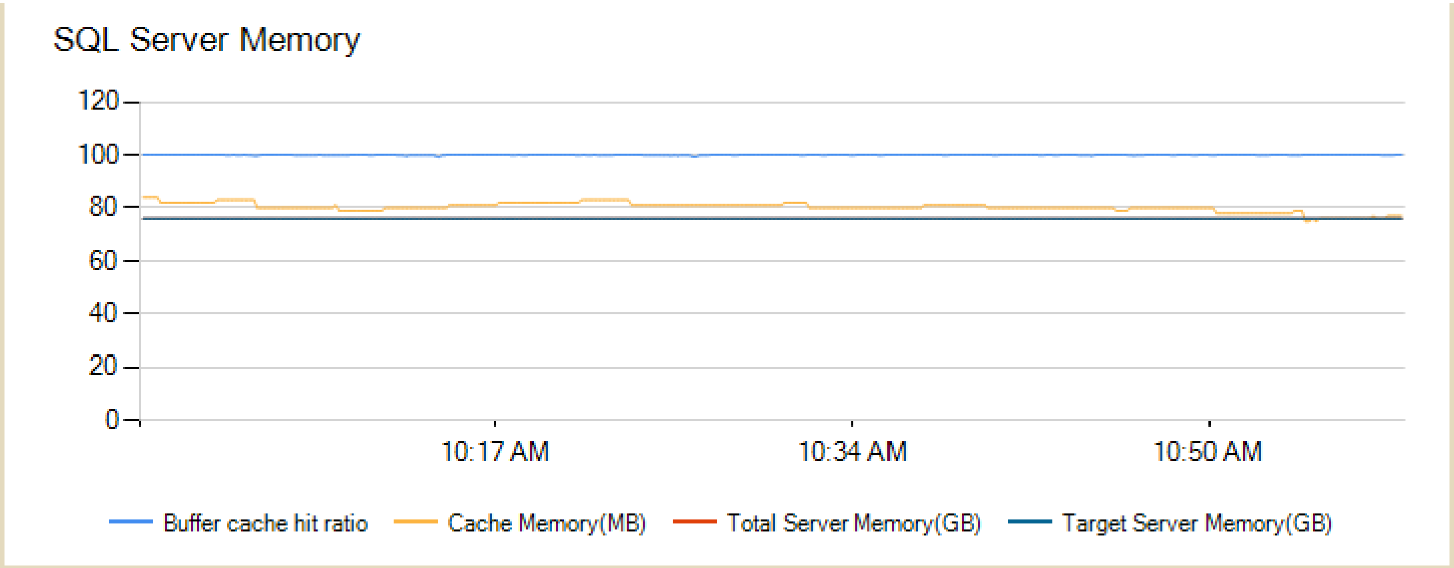
\SQLServer:Access Methods\Page Splits/sec

The SQLServer:Access Methods\Page Splits/sec counter measures the number of times SQL Server had to split a page when updating or inserting data per second. Page splits are expensive, and cause your table to perform more poorly due to fragmentation. The fewer page splits you have the better your system will perform.

Optimally, this counter should be less than 10%-20% of the batch requests per second.

Microsoft® Description – Number of page splits per second that occur as a result of overflowing index pages.

SQL Server Memory



\SQLServer:Buffer Manager\Buffer cache hit ratio

The buffer cache hit ratio counter represents how often SQL Server is able to find data pages in its buffer cache when a query needs a data page.

You want this number to be as close to 100 as possible. A low buffer cache hit ratio could indicate a memory problem.

Microsoft® Description – Percentage of pages that were found in the buffer pool without having to incur a read from disk.

\SQLServer:Memory Manager\SQL Cache Memory (KB)

The SQLServer:Memory Manager\SQL Cache Memory (KB) counter specifies the total amount of dynamic memory the server is using for the dynamic SQL plan cache.

If you see sudden drops over time for this counter, it might be an indication that the instance is under memory pressure and SQL Server had to reclaim part of the plan cache for other use by SQL Server, which causes this counter to suddenly decrease.

If you see sudden increases in this counter, this may indicate that a large number of one-time use ad hoc queries may have been executed, causing plan cache pollution. If this is a long term condition, consider turning on the "optimize for ad hoc workloads" instance-level option to stop plan cache pollution.

Microsoft® Description – Total amount of dynamic memory the server is using for the dynamic SQL cache

\SQLServer:Memory Manager\Total Server Memory (KB)

The SQLServer:Memory Manager: Total Server Memory (KB) tells you how much the mssqlserver service is currently using. This includes the total of the buffers committed to the SQL Server BPool and the OS buffers of the type "OS in Use".

Microsoft® Description – Total amount of dynamic memory the server is currently consuming

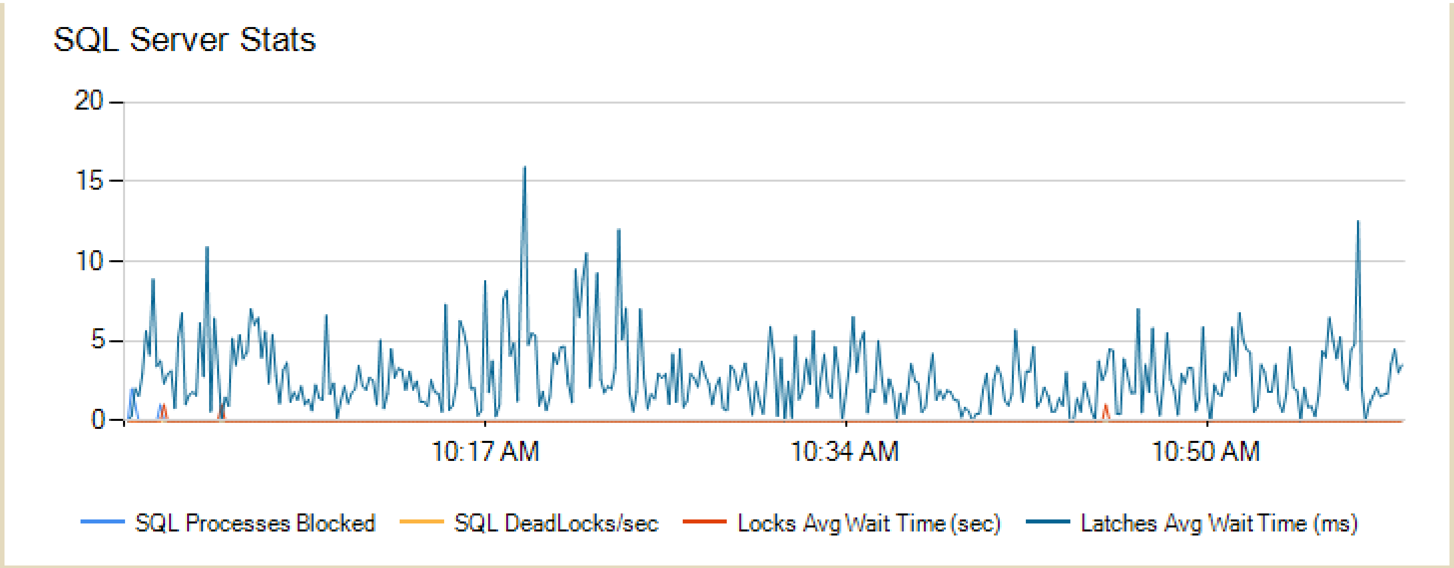
\SQLServer:Memory Manager\Target Server Memory (KB)

The SQLServer:Memory Manager: Target Server Memory (KB) counter tells you how much memory SQL Server would like to have in order to operate efficiently. This is based on the number of buffers reserved by SQL Server when it is first started up.

If the SQLServer:Memory Manager: Total Server Memory (KB) counter is more than the SQLServer:Memory Manager: Target Server Memory (KB) counter, this may indicate that SQL Server may be under memory pressure and could use access to more physical memory.

Microsoft® Description – Ideal amount of memory the server is willing to consume

SQL Server Stats



\SQLServer:General Statistics\Processes Blocked

The processes blocked counter displays the number of blocked processes.

When one process is blocking another process, the blocked process cannot move forward with its execution plan until the resource that is causing it to wait is freed up. Optimally you don't want to see any blocked processes

Microsoft® Description – Number of currently blocked processes.

\SQLServer:Locks(_total)\Number of DeadLocks/sec

The SQLServer:Locks(_total)\Number of DeadLocks/sec counter displays the number of lock requests per second that resulted in a Deadlock. Since this number is per second, this number should not be high.

Use the Number of Deadlocks/sec counter on a regular basis to get an idea of what appears normal and if you discover deadlock problems use the Profiler to investigate further.

Microsoft® Description – Number of lock requests that resulted in a deadlock.

\SQLServer:Locks(_total)\Average Wait Time (ms)

The SQL Server:Locks\Average Wait Time (ms) counter measures the average wait time of a variety of locks, including: database, extent, Key, Page, RID, and table.

You can use this counter to get a baseline value for the Average Wait Time on your servers, then investigate further if you see unexpected values.

Microsoft® Description – The average amount of wait time (milliseconds) for each lock request that resulted in a wait.

\SQLServer:Latches\Average Latch Wait Time (ms)

The SQLServer:Latches\Average Latch Wait Time (ms) counter measures the wait time (in milliseconds) for latch requests that have to wait (A latch is basically a lightweight lock).

This is a measurement for only those latches whose requests had to wait, not all latches. In many cases, there is no wait.

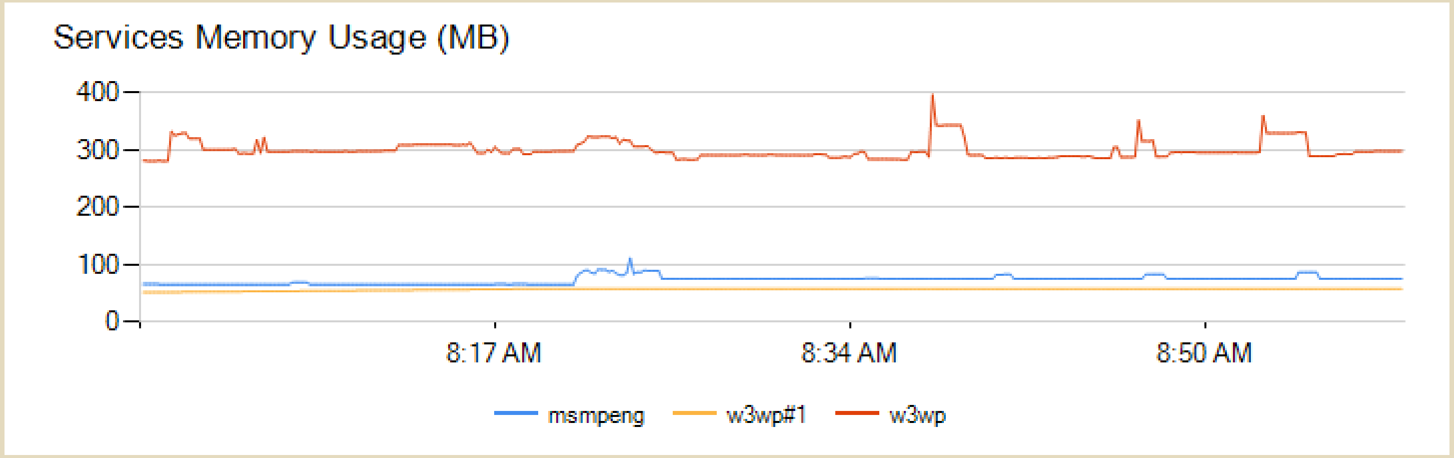
Microsoft® Description – Average latch wait time (milliseconds) for latch requests that had to wait.

Individual Process Statistics

Note: Our BLG Performance Report Generator can graph individual process stats while SPASS – System Performance and Sizing Site currently does not.

To give you the tools to drill down to see what process is consuming all the resources, we decided to add the ability to graph the stats from individual processes. Keep in mind that the more processes you select to generate the stats for, the longer the report will take to be generated.

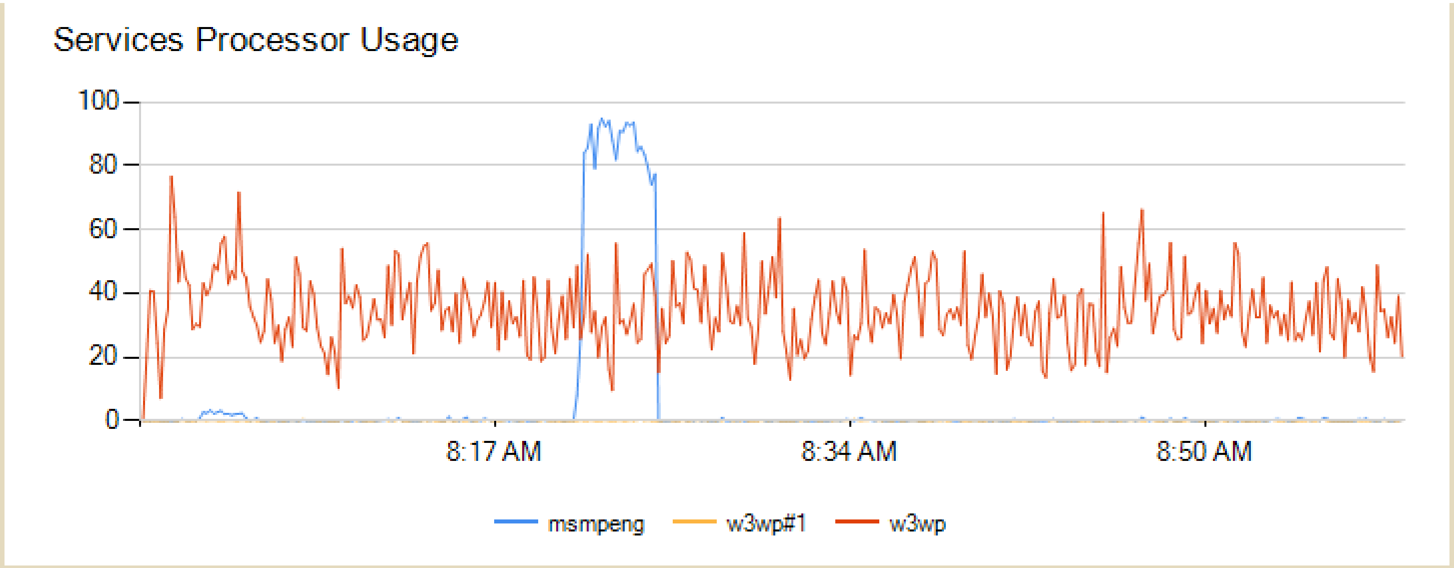
Also, there is a known issue that when you combine multiple logfiles using the relog utility (which is what we use) there may be un-realistic spikes in some counters. For instance CPU usage may go above 1000% or Memory Usage could go well above the the amount of memory in the system. Also, if there are multiple processes with the same name, when combining the logfiles the data may be crossed, if this is the case, the process graph line may suddenly change color since it might be showing under a different instance (i.e. w3wp#1, etc.).



\Process(processname)\Working Set

The Process(X)\Working Set shows the amount of physical memory the process is using.

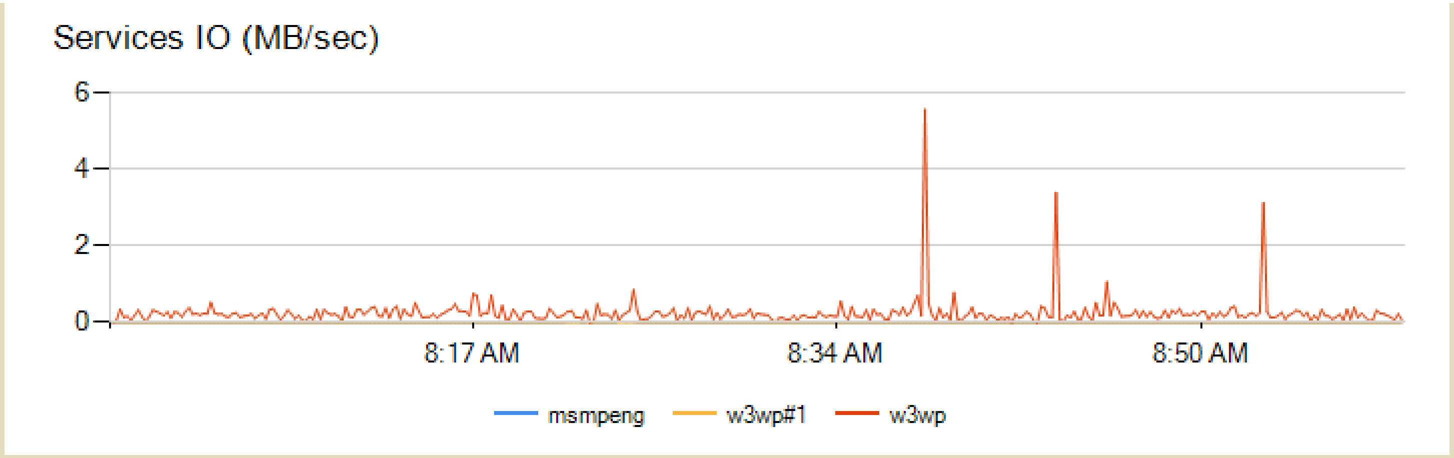
Microsoft® Description – Working Set is the current size, in bytes, of the Working Set of this process. The Working Set is the set of memory pages touched recently by the threads in the process. If free memory in the computer is above a threshold, pages are left in the Working Set of a process even if they are not in use. When free memory falls below a threshold, pages are trimmed from Working Sets. If they are needed they will then be soft-faulted back into the Working Set before leaving main memory.



\Process(processname)\% Processor Time

The Process(X)\% Processor Time shows the amount of CPU usage that the process is using, note that a multi-threaded process on a multi-core server could drive this counter above 100%.

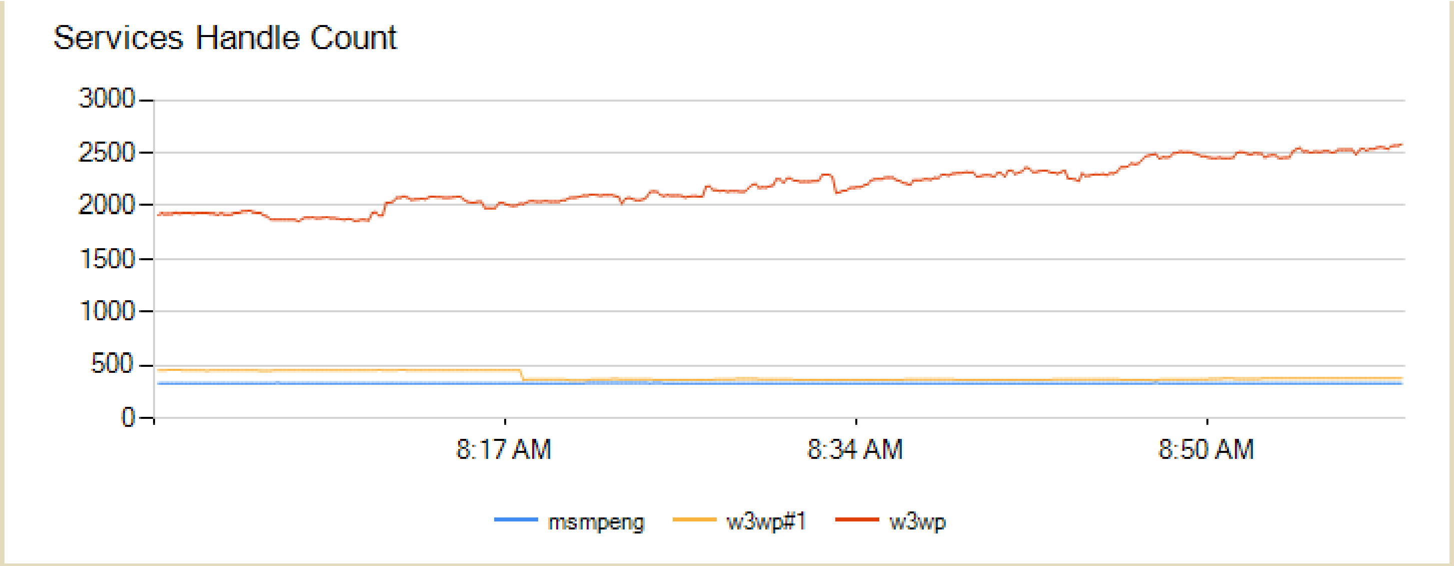
Microsoft® Description – % Processor Time is the percentage of elapsed time that all of process threads used the processor to execution instructions. An instruction is the basic unit of execution in a computer, a thread is the object that executes instructions, and a process is the object created when a program is run. Code executed to handle some hardware interrupts and trap conditions are included in this count.



\Process(processname)\IO Data Bytes/sec

The Process(X)\IO Data Bytes/sec counter shows the amount of I/O bytes the process is using. I/O consists of File Operations, Network Operations and Device Operations.

Microsoft® Description – The rate at which the process is reading and writing bytes in I/O operations. This counter counts all I/O activity generated by the process to include file, network and device I/Os.



\Process(processname)\Handle Count

Increased handle usage over time can identify certain memory leaks for some processes. With .net apps, this has become increasing rare, but it is still something to look for.

Microsoft® Description – The total number of handles currently open by this process. This number is equal to the sum of the handles currently open by each thread in this process.

Search