Establishing and Setting Thresholds
FOR A SQL SERVER ENVIRONMENT
BY ROBERT L. DAVIS
Summary:
This white paper is a guide for DBAs about determining the proper thresholds for monitoring a SQL Server environment.

Monitoring SQL Server is a critical component of a Database Administrator’s, or DBA’s, job. The bigger the environment is, the more important monitoring becomes. Enterprise environments typically have a high ratio of servers to DBAs and no DBA can keep watch of every aspect of the environment on their own. A very important aspect of monitoring is alerting.

Monitoring alone isn’t enough. When certain resource metrics or counters are reached, the DBA needs to be alerted and, in some cases, immediately notified that the threshold has been reached. Alerts can be delivered in a variety of ways and should not be confused with notifications. Alerts can be written to a log, raised in an application interface, or sent to another system for processing. A notification is a response to an alert that informs a person, or a group of people, that a specific alert has been raised.

Monitoring, alerts, and notifications all work together to form a good monitoring solution. Any good monitoring tool will perform these three basic functions. An additional function that is not part of all monitoring tools is baselining. Baselining is important for effective monitoring as most metrics or counters can vary greatly from system to system.

There are only a handful of counters that can unmistakably state that a specific value indicates a problem. Most counters have to be watched over time to establish an acceptable threshold for a healthy system. A baseline is a continual process. You can’t just baseline once. As systems evolve, so will their performance profiles. So a really good monitoring tool will have a built-in baselining functionality.
Why We Baseline

Having a good approach to troubleshooting is a lot like the methodologies used by a doctor. Unfortunately, a lot of people take the approach of offering up potential solutions without knowing what the core problem is, hoping that it fixes it. Imagine if you told your doctor that you had a sore throat and he gave you an assortment of pills, telling you to take the blue ones today and if you don’t feel better, take the red ones the following day. Hopefully, you’d go to a different doctor… one that would figure out what was wrong with you and treat it appropriately.

What’s the first thing they do in the doctor’s office when you see a doctor for the first time? They check your vital signs. They get your weight, height, blood pressure, pulse, temperature, and so on. Then when you come back on subsequent events, they check it again. Why do they do this?

The doctor is doing a lot more than just trying to justify the high cost of a doctor visit. They are getting a baseline of your health and then they are validating that baseline to see how your health is changing. In many ways, this baseline and check system may help them discover health issues that you aren’t even aware of yet. When I went in to see my doctor for a stomach flu, my blood pressure was a little higher than normal but not enough to be concerned. The slight elevation was probably due to my illness. If the next time I go, it is even higher or remains higher without the illness, then the doctor might get concerned. This is why they advise you to do regular physicals.

A regular physical will tell what my vital signs look like when I am healthy and no extenuating circumstances affects the numbers. Body temperature is a great statistic to use here. The normal temperature for a person is considered to be 98.6 degrees Fahrenheit (37.0 degrees Celsius). In actuality, the normal temperature is usually in the range of 97 to 100 degrees Fahrenheit (36.1 to 37.8 degrees Celsius), according to “The Physician’s Factbook”. So if I’m normally at the high end of the scale, and I come in with a temperature of 100 degrees, would the doctor say that I have a fever or not? Unless he’s done a baseline, he’d have no way to know if I was normal or running high. He’d look at past visits and physicals, if there were any, to see how I compared.

You need to do the same thing with your SQL Servers. So many times, all we hear are complaints that something is slow. How do you fix a problem like “slow”? You need to know what’s performing differently when it is slow as compared to how it normally runs. I don’t have enough fingers and toes to count how many things exist that can cause SQL Server to run slow. Heck, I’d need more hands just to count the things that could make disk I/O slow.

When somebody says that their SQL Server is running slow, I’ll ask, “What has changed?” Usually, they start talking about code changes when I really want to know what measurable statistic is different than normal. If someone says that full scans/second is high, I’ll ask what the normal scan rate is. The answer is usually “I don’t know.” If you don’t know what the normal scan rate is, how do you know it is high? Generally, when someone asks me about a specific counter, it is because they read somewhere that the counter shouldn’t be above a certain level. Lots of posts out there will tell you that full scans/second should be 0. That would be nice, but it’s not realistic. I’d be alarmed if the scan rate on one of my servers was 0 because that would mean that no users were able to hit the server.

For many performance counters, there is no definitive value that will indicate a problem. You have to know what the counter looks like when it is healthy to know when it is not healthy. If I have a baseline for reference, I can compare the current metrics with metrics from the day before, the week before, or some other time. There may even be times when I may want to compare it to much older baselines if there is some quarterly or annual event occurring. For example, if I managed a SQL Server that powered a retail website and we were in the holiday shopping season, I may want to compare the current performance to the metrics from one year ago.
How To Baseline

There are several tools you can use to help you baseline. There are a number of third party applications that will monitor your SQL Server, raise alerts, and send notifications for critical alerts. When selecting these tools, I consider built-in performance baselining a key feature. It is the easiest and fastest way to implement baselining.

If you’re the do-it-yourself type or don’t have a budget for one of these tools, you can implement some basic baselining with built-in Windows and SQL Server tools. Windows Performance Monitor (PerfMon) is a tool every DBA should know how to use. Setting up a PerfMon collection is pretty straight-forward. It can be used as your baseline collection tool, but does not provide any mechanism for raising alerts or sending notifications. It is only a small piece of the puzzle.

SQL Server does have the ability to raise alerts and send notifications. If you are doing it on your own, you will need to write your own process to check the performance counters and take the appropriate action if a threshold is crossed. The Dynamic Management View (DMV) sys.dm_os_performance_counters has the counters you will need to monitor SQL Server. You can poll the counters at regular intervals, check for metrics that exceed your defined thresholds and take action.

SQL Server has a built-in alerting system, but it has limited applicability. To be able to raise alerts and send notifications for all of the performance metrics you want, you will need to query the DMV noted above. SQL Server also has the management data warehouse (MDW) which has built-in collectors you can use to capture performance data from multiple servers and collect in a central store. The MDW comes with stock reports that can be used to view the performance information. It could be a useful part of baselining and historical reporting on performance, but it is generally not collected at a high frequency and does not raise alerts or send notifications. Also, if you choose to build alerting and notifications off of the MDW database, it will require custom coding.

Establishing and Setting Thresholds

Whatever method you decide to use for monitoring, baselining, and alerting, the tricky part is establishing and setting thresholds for the alerts. For many counters, this means utilizing baselines to determine thresholds to set for your counters. Even the counters that have recommended standard thresholds often require tweaking to adjust to differences in the system. The important thing is to identify what the counters look like when the server is healthy, and at what point the counters indicate an unhealthy server.

Described below are the counters that I find important to monitor and baseline, and are helpful when troubleshooting general server performance. Where plausible, I have provided thresholds for warning and critical alerts for the counters.

Disk Counters

One question that I hear occasionally is whether it is better to monitor logical disk counters or physical disk counters. With the exception of free space counters (percent and MB) in the logical disk group, you get the same set of counters with both sets. The key difference between them is in how the disks are grouped. If you isolate disks at the physical level, then each physical disk also represents a complete logical disk. It is common, however, to place the
C drive (system) and D drive (SQL install drive) on the same logical disk. If the C and D drive are partitions of the same physical drive, they will be the same physical drive but separate logical drives.

When you have logical drives sharing the same physical drive, you will likely want to know the counters at both the logical and physical level. If the disks are isolated, then either counter set will return the same data. In such a case, I generally prefer physical disk counters as they are labeled by drive letter rather than drive number.

Below you will find the disk counters I find useful and how I set thresholds for them. For the sake of brevity, I list physical disk counters only, but these also apply equally to logical disk counters.

- **Avg. Disk sec/Read:** Average time spent for each read. A constant high number in this metric may indicate slow disk system or a steady, heavy workload. Spikes in this metric may represent a disk system insufficient for sudden heavy loads.
  - Warning threshold of 10 ms.
  - Critical threshold of 20 ms.
  - This counter may be adjusted based on acceptable values determined through baselining.

- **Avg. Disk sec/Write:** Average time spent for each write. A constant high number in this metric may indicate slow disk system or a steady, heavy workload. Spikes in this metric may represent a disk system insufficient for sudden heavy loads.
  - Warning threshold of 10 ms.
  - Critical threshold of 20 ms.
  - This counter may be adjusted based on acceptable values determined through baselining.

- **Disk Reads/sec:** Average disk reads per second. I do not alert on this counter. I use it to gain more information when one of the per/second counters was high, had spiked, or changed.
  - This counter should be baselined and watched for changes.

- **Disk Writes/sec:** Average disk writes per second. I do not alert on this counter. I use it to gain more information when one of the per/second counters was high, had spiked, or changed.
  - This counter should be baselined and watched for changes.

- **Avg. Disk Bytes/Read:** Average size in bytes of reads. I do not alert on this counter. I use it to gain more information when one of the per/second counters was high, had spiked, or changed.
  - This counter should be baselined and watched for changes.
  - Changes in this counter can represent a change in the workload or may help identify inefficient read profile.
• **Avg. Disk Bytes/Write**: Average size in bytes of writes. I do not alert on this counter. I use it to gain more information when one of the per/second counters was high, had spiked, or changed.

• This counter should be baselined and watched for changes.

• Changes in this counter can represent a change in the workload or may help identify inefficient read profile.

You may have noticed that a very popular type of disk metric was completely ignored. There is no mention of any of the counters related to disk queue. The reason for this is that disk queue is very often misleading or misunderstood in modern systems. In order to properly understand disk queue metrics, you must know how many actual disks are behind the drive. Additionally, most SANs have built-in caching that hide disk queuing from the server-side.

**Memory or NUMA Node Memory**

A popular memory counter you hear people talk about is Page Faults/sec. A page fault is usually defined as a check in memory for a page that fails and results in a read from disk. However, this is only part of what the counter tells us. It counts both hard page faults (read from disk) and soft page faults (successful read from memory). All SQL Server requests for a read are measured as a page fault. So without more information, the counter is not very useful.

These counters are also available under the NUMA Node Memory counter object. You can measure these counters at individual NUMA nodes on NUMA systems. For production systems under heavy workload, it may be advantageous to view these counters at the NUMA level. You may encounter issues where the workload is uneven and one or a few NUMA nodes are under pressure when others are not.

• **Available Mbytes**: This counter is also available in Bytes and Kbytes. I monitor Mbytes because modern servers have a lot of memory. This counter can indicate memory pressure external to SQL Server.
  - A good starting point for a warning threshold is 10% of total memory.
  - A good critical threshold is 5% of total memory.
  - The counter should be baselined and sudden or drastic changes in available memory should be investigated.

• **Pages/sec**: This counter should be as low as possible. 0 is the preferable metric for this counter, but it may not always be possible.
  - This counter should be baselined and watched for changes. The baseline can help you determine some thresholds to use for alerting.
Network Interface

Network counters are tracked at the Network Interface Card (NIC) level. Below are four counters that I generally track and baseline for network activity. Each of these counters can vary greatly from one NIC card to another, even within the same machine. It depends on how they are used and for what purpose. Each counter should be baselined and watched for changes. These counters describe the amount of activity by each NIC card.

- **Bytes Received/sec**
- **Bytes Sent/sec**
- **Packets Received/sec**
- **Packets Sent/sec**

Additionally, I monitor for packets that get discarded and packets that generate an error. Both of these counters may indicate that packets are being dropped and should typically be 0. Numbers greater than 0 may indicate pressure on the NIC due to high activity (discards) or problems with the NIC or network (errors). I baseline these values, but only use them when connection issues are reported and I need to determine if there are issues at the NIC level.

- **Packets Received Discarded**
- **Packets Received Errors**
- **Packets Outbound Discarded**
- **Packets Outbound Errors**

Processor

Processors keep getting more complex with NUMA architectures (non-uniform memory access) and hyper-threading. Yet the way we monitor processor usage has remained relatively the same. Basically, we just need to know how busy it is and raise alerts when it gets above a certain level.

- **Avg % Processor Time:** This is the percentage of time that the processors are busy. I monitor this counter per processor as well as the total counter. Typically, I only alert on the total counter as one or a few busy processors on a server with many processors may not be indicative of a problem. Baselining should always include individual processors.
  - Warning threshold is 75%
  - Critical threshold is 90%
  - This counter should also be baselined and watched for changes. Thresholds may be adjusted based on baselined values.

- **Peak % Processor Time:** This is the highest usage that the processor experienced for the given interval. This can be indicative that the processor usage is having very high, very short spikes in activity. This counter simply gives you an idea of how high the usage went. I do not alert on this counter and only refer to it when investigating a potential processor bottleneck.
**SQL Server Counters**

- **SQLServer:Access Methods**
  - `Worktables Created/Sec`: Work tables are special areas in the tempdb space that are used to store temporary results such as spools, large object (Max, XML) variables, and cursors. This will vary widely, and I generally do not alert on this counter. Baseline this counter and watch for changes. I may also investigate this counter when tempdb activity is very high.
    - Thresholds can be set after a consistent baseline is collected, but I generally do not set alerts for this.
  - `Workfiles Created/Sec`: Work files are special areas in the tempdb space that are used to store temporary results such as hash joins and hash aggregates. This will vary widely, and I generally do not alert on this counter. Baseline this counter and watch for changes. I may also investigate this counter when tempdb activity is very high.
    - Thresholds can be set after a consistent baseline is collected, but I generally do not set alerts for this.
  - `Full Scans/Sec`: Full scans can be a major indicator of performance health, but the counter will vary widely by system. There is no way to determine from the counter if the scans are large or small. A scan of a 10 row table and a scan of a 10 million row table both count as 1 scan.
    - Thresholds should be determined by collecting a baseline to identify a healthy scan count
    - Sudden changes in this counter may or may not indicate a problem, but it warrants further investigation to determine if it does. It might be a change in workload that is not a problem for the system.
    - Use the DMV `sys.dm_db_index_usage_stats` to determine which indexes and tables are experiencing very high levels of scans. This could indicate that additional indexing is needed to alleviate large scans, or that the scans are not a problem.
  - `Page Splits/Sec`: Page splits, like page faults, is another tricky counter. All new allocations of pages are counted as a page split, even if splitting an existing page is not required. As a result, this number can be high even in a healthy system.
    - Baseline this counter and watch for changes.
    - If further investigation is warranted, Extended Events can be used to determine if the page splits are true page splits or just page allocations.

- **SQLServer Buffer Manager\Buffer Cache Hit Ratio**: Buffer cache hit ratio is the ratio of reads successfully read from the buffer cache divided by the total number of read attempts. You want this counter to be as close to 100 (percent) as possible, but a system can deviate from this a bit for short periods of time and still be quite healthy.
  - Warning threshold is 80%.
  - Critical threshold is 60%.
  - Baseline and watch for drastic dips in this counter.
• **SQLServer Buffer Node\Page Life Expectancy:** Page life expectancy (PLE) is a counter that is often misunderstood. There is a lot of information out there about specific numbers for these counters. Those recommendations are mostly outdated.

A PLE of 300 is an old recommendation that no longer makes sense. This was a good metric to alert on when SQL Servers had very small amounts of memory. The number you should use today varies greatly based on many factors, including the amount of memory available.

This counter can also be seen under Buffer Manager, but that counter represents the PLE as a whole. Most modern servers have a NUMA architecture, each node having local memory to itself. Under Buffer Manager, the PLE of each NUMA node is added and then divided by the number of nodes to get the average PLE of the server. Buffer Node provides PLE per NUMA node and allows to identify when there is one or a few NUMA nodes experiencing problems that might otherwise go undetected.

PLE is the estimated time in seconds a page that does not get reused will stay in memory. Baseline this metric and determine what the proper thresholds for your system are when it is healthy.

• **SQLServer Plan Cache\Cache Pages:** This is the section size of the buffer pool reserved for execution plans. This counter can vary widely from system to system, and I do not alert on it. Sudden changes can indicate a memory pressure or simply a change in the workload.

  ° Baseline this counter and watch for sudden dips or spikes.

• **SQLServer SQLStatistics\Batch Requests/Sec:** Batch requests per second simply indicates the amount of work SQL Server is doing. It is used to find out what a server is able to do during stress tests. Outside of stress testing, it is also used to track how hard SQL Server is working. The counter alone does not represent a problem, but it can be used to identify if the workload has changed when investigating other issues.

  ° Baseline this counter and watch for changes over time.

• **SQLServer Databases\Transactions/Sec:** Transactions per second simply indicates the amount of work SQL Server is doing. It is used to find out what a server is able to do during stress tests. Outside of stress testing, it is also used to track how hard SQL Server is working. The counter alone does not represent a problem, but it can be used to identify if the workload has changed when investigating other issues.

  ° Baseline this counter and watch for changes over time.

• **SQLServer General Statistics:** The temp table counters can vary wildly from server to server, from workload to workload, and even from one time of the day to another. These counters represent how quickly temp tables are being created and destroyed as well as the number of active temp tables that are in use at a given time. These counters alone do not indicate a problem; however, they can be useful when investigating heavy tempdb usage. Use these counters to determine the root cause of tempdb load when it is under pressure. These counters correlate directly to usage of temp tables and table variables in SQL code.

  ° \Active Temp Tables
  ° \Temp Table Creation Rate
  ° \Temp Tables For Destruction
Conclusion

Making the decision to move SQL Server to a virtual platform the first time can be a nervous time, but it doesn’t need to be. With proper planning and best practices to guide you, it is pretty straightforward to have a successful migration to virtual machines. You can use the methods laid out in this paper to wade into virtualization slowly with noncritical workloads and small footprint servers before moving on to considering critical production workloads.

You can move SQL Server to a virtual platform with ease using these guidelines. Don’t make the same common mistakes that others have made. Use the best practices outlined here to guide you and help you avoid these pitfalls. The licensing information provided should make the complex licensing rules a lot clearer and help you make the right licensing decisions for virtualizing SQL Server.

About the Author:

Robert L Davis is a senior database administrator and technical lead at Microsoft. He is a speaker and a trainer as well as a writer for SQL Server Magazine, & co-authored “Pro SQL Server 2008 Mirroring” (Apress). Blog: www.sqlsoldier.com Twitter @sqlsoldier

Idera was started in 2004 and is now the leader in application and server management software. Their over 12,000 global clients include Fortune 1,000 enterprises, government entities, and several of the world’s largest datacenters and cloud infrastructure providers. These and many other companies rely on Idera’s products to keep data backed up and protected, while keeping their servers optimized for performance.

Idera’s products help keep servers running smoothly, which potentially adds value to business. Continual server performance and efficiency keeps companies running at their peak, while allowing those who manage them to focus on major issues as they occur. Whether a company has ten servers or ten thousand, Idera and its products gives companies of all sizes the best performance and value with products that are easy to install and deploy to deliver real and measurable ROI.

Headquartered in Houston, Texas, Idera is a Microsoft Managed Partner with international offices in Europe, Asia Pacific, and Latin America.
TRY IDERA’S SQL DIAGNOSTIC MANAGER

24x7 SQL PERFORMANCE MONITORING, ALERTING AND DIAGNOSTICS

- Performance monitoring for physical and virtual SQL Servers
- Deep query analysis to identify excessive waits and resource consumption
- History browsing to find and troubleshoot past issues
- Adaptive & automated alerting with 100+ pre-defined and configurable alerts
- Capacity planning to see database growth trends and minimize server sprawl
- SCOM management pack for integration with System Center

TRY IT FREE FOR 14 DAYS.