<http://www.brentozar.com/archive/2006/12/dba-101-using-perfmon-for-sql-performance-tuning/>

Setting Up Perfmon for SQL Server Tuning

On your desktop or laptop (not the SQL Server itself), go into Control Panel, Administrative Tools, Performance. The first thing that comes up is a 1990′s-looking line chart with a few basic performance stats on your workstation itself. Interesting, but not what we’re after.

On the left side, expand Performance Logs and Alerts, and click on Counter Logs. Counter Logs let us pick a set of performance counters and log them to a file periodically. One counter log is listed by default, but we’re going to add a new one. Right-click on Counter Logs and click New Log Settings. Name it with the name of your database server, because each server should have its own log settings. We could theoretically build a single counter log for all of our database servers, but then anytime we want to run the counter log against one of our servers to test its performance, it will log data for ALL servers, and that’s not usually how we want to do performance tuning. After typing in a name, we can start setting up the counters we want to log.

The Basics of SQL Server Perfmon Counters

Click the Add Counters button, and change the computer name to point to the SQL Server name instead of your workstation’s name. (If you have multiple SQL Server instances on the machine, don’t worry about it, that doesn’t apply here.) After typing in the server name, hit Tab, and the workstation will pause for a few moments. It’s gathering the list of performance objects available on that server. Each server will have different lists of performance objects depending on what software is installed on that server: for example, SQL Server 2005 offers a different set of counters than SQL Server 2000.

In the Performance object dropdown, choose the “Memory” object. The list of counters will change. Select the “Pages/sec” counter, and click the Add button. It will seem like nothing happened, but try clicking the Add button again. You’ll get an error saying that the counter was already added. It’s not exactly an elegant user interface, but it works. Technically. Now scroll up to the “Available MBytes” counter, highlight it, and click Add. Those are the only two memory counters we’re going to monitor for now.

In the Performance object dropdown, choose the “Processor” object, and in the counters list, highlight the “% Processor Time” counter. Notice that in the right hand window, we now have more instances to choose from. We can track the % Processor Time statistic for each individual processor, or for all of them combined (\_Total). Personally, I like to highlight each individual processor, and then click Add. I don’t find the \_Total statistic useful because it’s simply an added number combined from all of the individual processors. On a 4-CPU box (single cores, no hyperthreading), that means our \_Total statistic can be from 0 to 400. However, 100 can mean two different things: it could mean that each processor is running 25% utilization, or it could mean that one processor is pegged at 100% while the rest are twiddling their thumbs waiting for work. It could also mean any other number of combinations, like two processors at 50%. Therefore, the \_Total number usually gives me more questions than answers, so I don’t bother logging it. Highlight all of the processor instances except \_Total, and click the Add button.

In the Performance object dropdown, choose Physical Disk, and choose the “% Disk Time” counter. Notice that again on the right side window, we get multiple instances; this time, we get one per physical disk. In performance terms, physical disk means one disk shown in Computer Management’s Disk Management tool. One physical disk may have multiple partitions, each with its own drive letter, but for performance tuning, we want to know how hard that one physical disk is working.

This one “physical disk” may have a bunch of actual physical drives, like in RAID systems. However, Windows isn’t quite smart enough to know exactly how many drives are in the RAID array, so the term “physical disk” is a little misleading here.

Highlight all of the physical disks in the instance list (again, leave off the \_Total instance) and click the Add button. Then, in the counter list, choose the “Avg. Disk Queue Length” counter and add it too.

The Best SQL Server Performance Monitor Counters to Analyze

Now that you’ve got the hang of adding counters, here’s the full list we need to add, including the ones mentioned above:

* These are listed OBJECT first, then COUNTER
* Memory – Available MBytes
* Paging File – % Usage
* Physical Disk – Avg. Disk sec/Read
* Physical Disk – Avg. Disk sec/Write
* Physical Disk – Disk Reads/sec
* Physical Disk – Disk Writes/sec
* Processor – % Processor Time
* SQLServer: Buffer Manager – Buffer cache hit ratio
* SQLServer: Buffer Manager – Page life expectancy
* SQLServer: General Statistics – User Connections
* SQLServer: Memory Manager – Memory Grants Pending
* SQLServer: SQL Statistics – Batch Requests/sec
* SQLServer: SQL Statistics – Compilations/sec
* SQLServer: SQL Statistics – Recompilations/sec
* System – Processor Queue Length

After adding those, click the Close button, and we’re back to the counter log setup window. Under “Sample data every”, the default should be 15 seconds, and that’s fine. In the “Run As” entry, type in your domain username in the form domainname\username, and click the Set Password button to save your password. This lets the Perfmon service gather statistics using your domain permissions – otherwise, it tries to use its own credentials, and they probably won’t work on the remote server.

Click on the Log Files tab, and change the log file type to Text File (Comma delimited). This lets us import it into Excel easier. Click on the Configure button, and set the file path. I save mine in a file share called PerformanceLogs so that I can access it remotely via a mapped drive, and so that I can share it with other users who want to know how their server is performing. Click OK, and click OK again, and you’ll be back at the main Performance screen listing the counter logs.

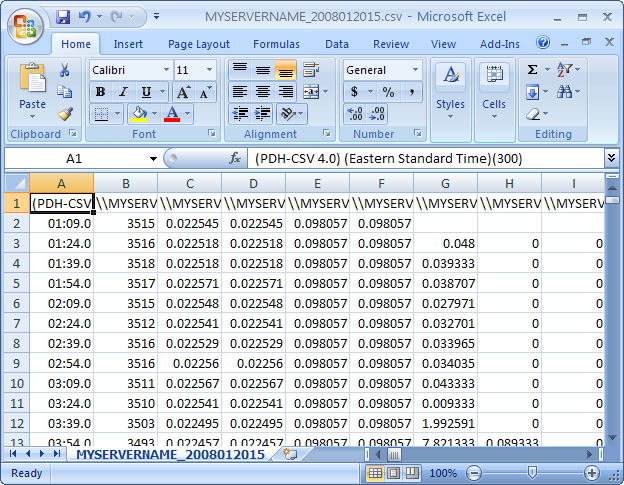
If all went well, your counter log is now running. To make sure, right-click on your new counter log’s name, and the Start option should be grayed out. If it’s not, most likely there was a permissions problem with the username & password you used. Click Properties, and try setting the username & password again.

Let Perfmon run for a day or two to gather a good baseline of the server’s activity. It’s not that invasive on the SQL Server being monitored, and the in-depth results will pay off. The more data we have, the better job we can do on analyzing the Perfmon results.

Turning SQL Server Perfmon Logs Into Pretty Excel Spreadsheets

After Perfmon has run for a day or two, go back in and right-click on the counter log and click Stop. That will cease monitoring of the database server.

Open the results CSV in Excel. It’s going to look ugly at first:



Perfmon Counters In Excel

This is the data from Perfmon. Each column represents one of the metrics, and each row represents one time period. It’s hard to see that with the way it’s currently formatted, but we’ll solve that. Time to put some lipstick on this pig. We’re going to:

* Get rid of the half-empty row #2 and cell A1
* Format the dates
* Format the numbers
* Clean up the headers
* Add formulas across the top (my Secret Sauce part)

Excel pros can figure out how to do this, but for the folks who don’t spend too much time in spreadsheets, I’ll explain how:

**Getting Rid of the Half-Empty Row and Cell A**Row #2 in this screenshot only has numbers in the first few columns. That’s typical for Perfmon – when it first starts monitoring a server, it takes it a round of checks before it’ll gather all of the data. Highlight row 2 and hit the Delete key on your keyboard. (I figured we’d start easy.)

Then delete the contents of cell A1, which we don’t need.

**Format the Dates in Column 1**

Click on cell A2 and look at the formula bar, and you’ll see that it’s a date – it’s just that Excel picks the wrong format to display it. Highlight all of column A (by clicking on the “A” button at the top of the column). Right-click on that A button and click Format Cells. Pick a Date format that includes both the date and the time, and click OK.

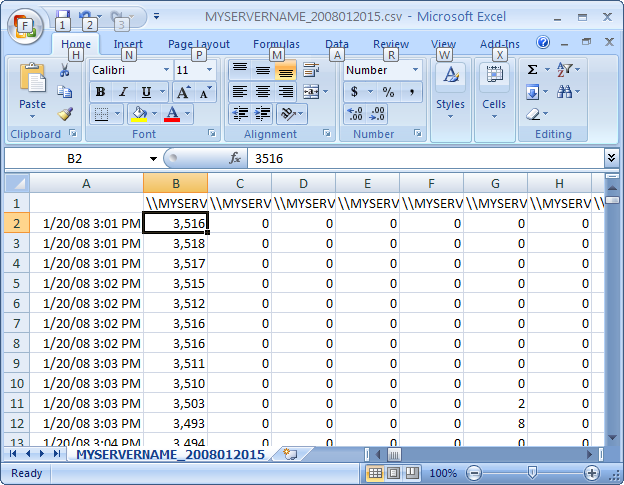
At that point, the column will probably show all “#######” entries, which just means it’s not wide-enough. Double-click on the line between the A and B columns, and Excel will autosize the column.

**Format the Numbers**

In our spreadsheet, some of the numbers have tons of stuff after the decimal place and it’s hard to read. Highlight all of the columns except A (our dates), and right-click and Format Cells. Choose the Number format, zero decimals, and check the box for Use 1000 Separator.

Some of our numbers will have valid data less than zero, but we’ll fix that later. Right now we’re just getting the basics cleaned up.

At this point, your spreadsheet should look like this:



Perfmon Results Formatted

Now let’s solve that mystery row of headers.

**Cleaning Up the Headers**

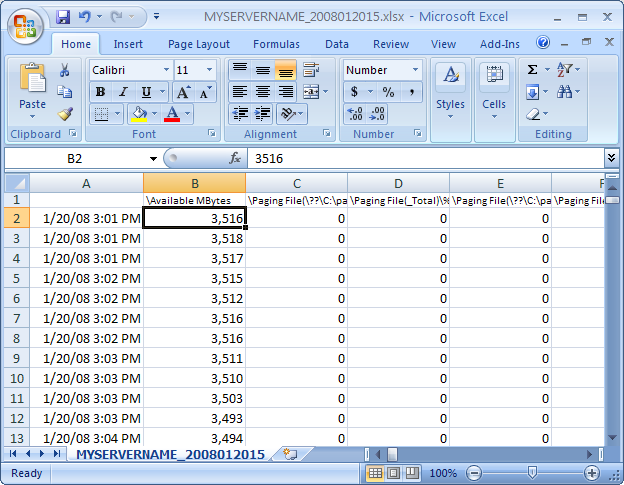
In row #1, notice how all of the cells in my screenshot start with \MYSERV. That’s because the server is called MYSERVERNAME. (Well, not really, but I faked it for this demo.) Click on one of those cells, and you’ll see the full text that Perfmon saves in the header, like “\MYSERVERNAMEMemoryAvailable MBytes”.

Do a find & replace by hitting Control-H. In the “Find What” box type your server name with two backslashes, like \MYSERVERNAME. In the “Replace with” box, erase everything. We want to just delete everywhere that says \MYSERVERNAME. Click Replace All, and Excel should tell you how many replacements it made.

Now do the same for these strings, but be careful to note which ones have spaces:

* Memory
* Paging File
* PhysicalDisk
* Processor
* SQLServer:Buffer Manager
* SQLServer:General Statistics
* SQLServer:Memory Manager
* SQLServer:SQL Statistics
* System

Now you’ll be able to tell a little more about what’s in each header, but to make it better, change the font size to 8 for that header row. Then highlight all of the columns and make them wider – not terribly wide, but just wide enough that you can see most of the headers for easier analysis. It’ll end up looking like this:

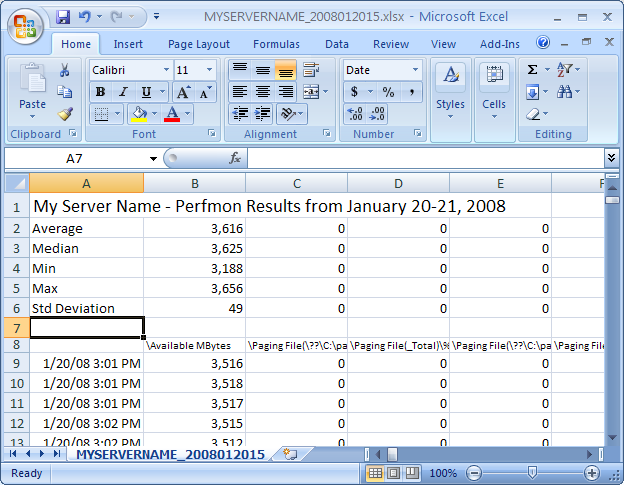


Perfmon Counter Names Now Visible

Getting better. Before the next step, go ahead and save the spreadsheet in Excel format, because the calculations will be slow.

**Add Formulas Across the Top**

Here comes the secret sauce: we want formulas across the top to quickly summarize our findings. It’ll be easier to understand by looking at the final desired result first:



Perfmon Results with Formulas

In this particular example, don’t be fooled by the zeroes in columns C through E – that’s because this server didn’t actually have any activity for those metrics.

Start by right-clicking on the Row 1 button at the far left of the screen and click Insert, and repeat that process seven times so that we’ve got seven empty rows at the top.

Then type in the labels in cells A2-A6 to match my screen shot.

In the B column, put in the formulas. In my typed examples below, I assume that your data goes from row 9 to row 100, but you’ll want to change that 100 number to wherever your last data row is.

* B2 (Average) will be =AVERAGE(B9:B100)
* B3 (Median) will be =MEDIAN(B9:B100)
* B4 (Min) will be =MIN(B9:B100)
* B5 (Max) will be =MAX(B9:B100)
* B6 (Std Deviation) will be =STDEV(B9:B100)

Finally, the icing on the cake: hit Control-Home, then move your cursor to cell B9, which should be your first cell with data. Hit Alt-W-F-Enter, which should Freeze Panes. That way we can move around our spreadsheet while we still see the dates on the left and the formulas & headers across the top.

Now, we’ve got a gorgeous spreadsheet that’s easier to roam around and analyze.

Analyzing the Performance Monitor Results to Find SQL Server Bottlenecks

Now for the detective work! There are a lot of smart people out there who have some great ways of interpreting Perfmon results. My favorite is the Microsoft SQL Server Customer Advisory Team (SQLCAT), who published two fantastic blog posts that sum up what counters to look at, and what they mean:

* [Top 10 SQL Server 2005 Performance Issues for Data Warehouse and Reporting Applications](http://sqlcat.com/top10lists/archive/2007/11/21/top-10-sql-server-2005-performance-issues-for-data-warehouse-and-reporting-applications.aspx)
* [Top SQL Server 2005 Performance Issues for OLTP Applications](http://sqlcat.com/top10lists/archive/2007/11/21/top-sql-server-2005-performance-issues-for-oltp-applications.aspx)

So now that we’ve established how the smart people do it, I’ll tell you how I do it, ha ha ho ho.

**Look for Obvious CPU Bottlenecks**

First, look at the Processor Queue Length for CPU pressure. If this number is averaging 1 or higher (except during the SQL Server’s full backup window if you’re using backup compression), this means things are waiting on CPUs to become available.

I’m suspicious when this number is over 1, because it often means that people have installed other software on the SQL Server such as applications or web sites. That’s a problem. If you get pushback from management saying that they don’t want to buy new servers, point out that two CPUs of SQL Server Enterprise licensing cost around $50-60k – which would pay for a separate server for the web app. If you can eliminate applications from the SQL Server, then you don’t have to use as much CPU power, and less CPUs mean less licensing costs.

There are more in-depth Perfmon metrics that you can add to your capture if you see the Processor Queue Length showing up, but for junior DBAs, the first thing I would recommend is simply remote desktop into the SQL Server. Right-click on the taskbar, click Task Manager, and click on the Processes tab. Check the box that shows processes for all users, and then click on the CPU column to sort the CPU percentage from high to low. Sit and watch it for a minute or two. Which processes are using CPU power? If it’s SQLServer, then we need to do more research, but there’s a good chance it’s another application, and we need to get that app off this server. Generally speaking, enabling hyperthreading on a SQL Server is not going to fix that kind of a problem.

**Look for Obvious Memory Problems**

Look at the % Usage metric, which monitors how much of the page file Windows is using. Generally speaking, you don’t want to see a SQL Server swapping memory out to disk. If this number is averaging 1% or more, then this server would benefit from more memory or setting SQL to use less memory.

1% doesn’t mean that money should be spent on more memory right away: in cost-sensitive shops, every server hits the page file sooner or later. It just points to a possible issue, and keep an eye on it.

On the Memory – Available MBytes statistic, look for fluctuations of a couple hundred megabytes or more. If that’s happening, then either the SQL Server’s memory is being adjusted dynamically (probably a bad idea for performance) or users are actively logging into the SQL Server via remote desktop and running software. Correlate these fluctuations with disk activity: when available memory drops, is disk activity also increasing? Is this disk activity affecting the page file drive? If so, use this as a demonstration to people using remote desktop – show them that this is why people shouldn’t remote desktop into database servers.

If the Memory – Available MBytes dips below 100mb, that’s an indication that the operating system may be getting starved for memory. Windows may be paging out your application to disk in order to keep some breathing room free for the OS.

**Look at Disk Metrics Last**

The reason we don’t look at disk performance metrics first is because memory problems can trigger disk problems. If a SQL Server doesn’t have enough memory, or if the SQL Server account doesn’t have the necessary permissions to lock pages in memory, then disk activity may be artificially high on the page file drive.

As we examine the disk numbers, make a mental note of which drive has the page file – and there may be multiple page files as well. Also, find out if one physical drive array is shared by multiple drive letters. This is especially important for servers that boot from SAN: the boot drive may contain the page file, and that boot drive may be on a shared set of spindles with several drive letters. Heavy page file activity will slow down access to all of the affected drive letters in Windows.

Another caveat: mount points. If the shop uses mount points, a drive letter may be broken up into several sets of physical drive arrays. If you don’t know what a mount point is, then you’re probably not using them, and you can ignore that advice.