[**Kev's Blog**](http://kswain.blogspot.com/)

**Saturday, April 12, 2008**

**[The “sys.dm\_os\_performance\_counters” Dynamic Management View](http://kswain.blogspot.com/2008/04/sysdmosperformancecounters-dynamic.html)**

**The "sys.dm\_os\_performance\_counters" Dynamic Management View**

By : Greg Larsen

SQL Server 2005 performance can be tracked and monitored by using performance counters. The performance counters can be displayed by either the System Monitor (PERFMON) tool, or by using the "sys.os\_exec\_performance\_counters" Dynamic Management View (DMV). This DMV is new with SQL Server 2005. This article will explore the different counters exposed by this DMV and how you can use this DMV to monitor various performance aspects of SQL Server.

**sys.os\_exec\_performance\_counters DMV**

The "sys.os\_exec\_performance\_counters" DMV, lets you use simple TSQL to obtain different SQL Server performance counters. This view contains both instance level and database specific counters. Some counters provided valuable information by themselves, while other counters require you to compare the difference between multiple counters, to obtain a meaningful counter value. Below is list of the different SQL Server objects available within this DMV. For each of these objects multiple counters exist:

SQLServer:Buffer Partition

SQLServer:User Settable

SQLServer:Databases

SQLServer:CLR

SQLServer:Cursor Manager by Type

SQLServer:Exec Statistics

SQLServer:Transactions

SQLServer:Memory Manager

SQLServer:SQL Errors

SQLServer:Buffer Node

SQLServer:Plan Cache

SQLServer:Access Methods

SQLServer:Cursor Manager Total

SQLServer:Broker Activation

SQLServer:Latches

SQLServer:Wait Statistics

SQLServer:Broker/DBM Transport

SQLServer:General Statistics

SQLServer:SQL Statistics

SQLServer:Catalog Metadata

SQLServer:Broker Statistics

SQLServer:Locks

SQLServer:Buffer Manage

As you can see, there are quite a few SQL Server objects that contain performance counters, which are exposed by this DMV. For more information about each of these objects, please refer to the Books Online topic titled "Using SQL Server Object".

These performance counters were available in SQL Server 2000 by retrieving information from the "master.dbo.sysperfinfo" table. A view that represents this table has been provided in SQL Server 2005, for backwards compatibility with SQL Server 2000. This view allows your old SQL Server 2000 code to still work and retrieve information about performance counters. But as with any backwards compatible feature you should consider re-writing your code to use the new "sys.os\_exec\_performance\_counters" DMV.

**Examples of how to use sys.dm\_os\_exec\_performance\_counters DMV**

The "sys.dm\_os\_exec\_performance\_counters" DMV can be used to retrieve a number of different SQL Server performance counters. Depending on what performance information you are trying to retrieve will determine which counters you should use. In order for you to better understand how to use this DMV let me go through a few examples.

For the first example, let me show you how to use this DMV to calculate the buffer cache hit ratio. The buffer cache hit ratio value measures how well your pages are staying in the buffer cache. The closer the buffer cache hit ratio is to 100 % the better your buffer cache is being utilized, and the better performance you will get out of your SQL Server. Here is how you would use this DMV to calculate the buffer cache hit ratio:

SELECT  
(a.cntr\_value \* 1.0 / b.cntr\_value)  
\* 100.0 [BufferCacheHitRatio]

FROM  
(SELECT  
\*, 1 x FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Buffer cache hit ratio'

AND  
object\_name  
=  
'SQLServer:Buffer Manager') a

JOIN

(SELECT  
\*, 1 x FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Buffer cache hit ratio base'

and  
object\_name  
=  
'SQLServer:Buffer Manager') b

ON a.x = b.x ;

Here I have used two different counters ("Buffer cache hit ratio" and "Buffer cache hit ratio base") of the "SQLServer:Buffer Manager" counter object. I used these counters to calculate the buffer cache hit ratio as a percentage. The counter name "Buffer cache hit ratio base" counts the number of time a query tries to find a page in the buffer cache, where as the "Buffer cache hit ratio" counter identifies the number of time a page was actually found in the cache. So to calculate the buffer cache hit ratio you simply divided the cntr\_value for the "Buffer cache hit ratio" counter by the "Buffer cache hit ratio base" cntr\_value and then multiply by 100 to represent the buffer cache hit ratio as a percentage. I joined the two different counter records together, in the FROM clause, so I could perform this calculation in a single T-SQL statement.

The "sys.dm\_os\_exec\_performance\_counters" DMV has a number of counters that require two different counter values to obtain a single meaningful performance number, like I performed in the above query. One of the counters of the pair is known as the base for the other counter. If you try to just use one of these counters that also has a base counter, like "Buffer cache hit ratio" it doesn't give you the true value performance measurement unless you divide the counter by the base. You can identify the counters that have a base by using the cntr\_type. A counter that requires a base to calculate a useful performance number will have a cntr\_type of "537003264". The base counters have a cntr\_type of "1073939712". The counter and it base have similar names, the only difference is the base counter name has the word "base" appended to the name of the counter. Make sure whenever you use a counter that has a base that you use the two counters together to calculate the "true" counter value.

Some of the counters in the "sys.dm\_os\_exec\_performance\_counters" DMV are known as "per second" counters. Counters of this type have "/sec" as part of their counter name, like "Page lookups/sec", or "Page writes/sec". The values for these counters are cumulative. To determine the actual per second value you need to capture the counter value at 2 different time periods and then calculate the difference in those values based on the time between the two samples. Here is an example where I have calculated the "Page reads/sec" by taking two samples:

DECLARE @time1 DATETIME;

DECLARE @time2 DATETIME;

DECLARE @value1 BIGINT;

DECLARE @value2 BIGINT;

-- get first sample

SELECT @value1=cntr\_value, @time1 =  
getdate()

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Page reads/sec';

-- wait for 10 seconds

WAITFOR  
DELAY  
'00:00:10';

-- get second sample

SELECT @value2=cntr\_value, @time2 =  
getdate()

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Page reads/sec';

-- calculate page reads per second

SELECT  
(@value2 - @value1)  
/  
datediff(ss,@time1,@time2) [Page reads/sec];

In this example I first captured the value of the "Page reads/sec" counter and the time the sample was taken and stored that information in 2 different local valuables, @value1, and @time1. I then waited for 10 seconds using the "WAITFOR" statements. Then I captured the second value for "Page reads/sec" counter and the time by setting the @value2, and @time2 variables. Lastly I calculated the difference between the two different counter values I collected, and divided that by the length of time between the two samples. By doing this I display the actual "Page reads/sec" that occurred between my two sampling intervals. In order to accurately determine the value for any of the "…/sec" counters you will need to capture two samples and then calculate the actual per second value.

This DMV also has counters that are tracked at the database level. Some of these database level counters are also rolled up into instance level counters (instance name = "\_Total"), one such counter is "Log Flushes/sec". For these types of counters you can look at the individual counters for each database, or sum all the database specific counters, and it will match the rolled up counter for the instance\_name "\_Total" counter. Below is an example where I reviewed the individual database counters for the "Log Flushes/sec" counter, then summed up the individual database counters to match the instance level "\_Total" counter:

SELECT  
\*

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Log Flushes/sec'  
AND instance\_name <>  
'\_Total';

SELECT  
sum(cntr\_value)

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Log Flushes/sec'  
AND instance\_name <>  
'\_Total';

SELECT  
\*

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Log Flushes/sec'  
AND instance\_name =  
'\_Total';

This DMV can even track SQL Server memory usage. Here are a couple of memory related counters worth monitoring: "Total Server Memory (KB)" and "Target Server Memory (KB)". The "Total Server Memory (KB)" counter is not an indicator of how much total memory SQL Server is using, but instead this counter only tracks how much memory SQL Server has used up for the buffer pool. The "Target Server Memory (KB)" counter indicates the maximum amount of dynamic memory SQL Server can use. Here is a query that returns these memory counters:

SELECT  
\*  
FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Total Server Memory (KB)'

OR counter\_name =  
'Target Server Memory (KB)';

I would suggest you run this query periodically to get an idea of how your SQL Server is managing memory. You will find that after SQL Server has been up for a while the "Total Server Memory (KB)" counter will be the same as the "Target Server Memory (KB)" counter.

Below is a query that returned the Page Life Expectancy counter:

SELECT  
\*  
FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Page life expectancy'

AND  
object\_name  
=  
'SQLServer:Buffer Manager';

The "Page life expectancy" counter indicates the average number of seconds a page will stay in the buffer cache. A good rule of thumb for how long a page should stay in the buffer cache for an Online Transaction Processing (OLTP) application is 300 seconds, or 5 minutes. If your page life expectancy is below this number then you might be experiencing some memory issue. This issue could be caused by because you are missing some key indexes, or having a buffer cache flush problem. You should periodically check the "Page life expectancy" number so you can better understand how many seconds your pages are staying in the buffer cache. If you suddenly see a drop in the "Page life expectancy" value then you could potentially be having some SQL Server memory issues.

If you want to monitor how fast your backups or restores are writing or reading I/O from your backup devices you can use the "Backup/Restore Throughput/sec" counter. Once again this is a per second counter so you need to take 2 samples of the counter value and compare them to get the actual throughput per second. This counter measures the number of bytes read or written to a backup device. Below is an example of how to calculate the throughput of a backup for the AdventureWorks database. Keep in mind you need to run this query when a backup is running otherwise it will return 0 for a throughput value:

DECLARE @time1 DATETIME;

DECLARE @time2 DATETIME;

DECLARE @value1 BIGINT;

DECLARE @value2 BIGINT;

-- get first sample

SELECT @value1=cntr\_value, @time1 =  
getdate()

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Backup/Restore Throughput/sec'

AND instance\_name =  
'AdventureWorks';

-- wait for 10 seconds

WAITFOR  
DELAY  
'00:00:05';

-- get second sample

SELECT @value2=cntr\_value, @time2 =  
getdate()

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Backup/Restore Throughput/sec'

AND instance\_name =  
'AdventureWorks';

-- calculate page reads per second

SELECT  
(@value2 - @value1)  
/  
datediff(ss,@time1,@time2) [Backup/Restore Throughput/sec];

If your backups are taking a long time to complete, then this measurement can be used to determine how many bytes per second are being written to your backup device. The higher the number the better. If you want to experiment with adding additional backup devices or faster backup devices you can use this measurement to determine if your changes have improved or hurt your backup throughput.

To calculate the amount of Log space you could used the output of the DBCC SQLPERF(logspace) command. The "sys.dm\_os\_performance\_counters" DMV also provided a method to obtain the amount of Log space used for a given database. Here is an example that returns the MB of Log space used by the AdventureWorks Databases:

SELECT cntr\_value / 1024.0 [Log Space Used (MB)]

FROM  
sys.dm\_os\_performance\_counters

WHERE counter\_name =  
'Log File(s) Used Size (KB)'

AND instance\_name =  
'AdventureWorks'

**Conclusion**

As you can see the "sys.dm\_os\_performance\_counters" DMV allows you an easy method to get at SQL Server performance counters using a simple SELECT statement. You can use this DMV to develop a set of queries that you periodically run to monitor performance counters. By periodically monitoring these counters you will get a sense for how your application uses resources and the affect they have on the performance counters. By routinely monitoring performance counters, and reviewing the counters to see if they are similar to your last counter snapshot, you can quickly identify if your server is having performance issues.

[http://img2.blogblog.com/img/icon18_edit_allbkg.gif](http://www.blogger.com/post-edit.g?blogID=7990571641413100676&postID=1010777355613278338)