**SQL Server queries with DMVs for examining bottlenecks**

SearchSQLServer.com

* [Digg This](http://digg.com/submit?phase=2&url=http://searchsqlserver.techtarget.com/tutorial/SQL-Server-queries-with-DMVs-for-examining-bottlenecks&title=SQL%20Server%20queries%20with%20DMVs%20for%20examining%20bottlenecks&topic=tech_news&bodytext=Here,%20you'll%20find%20queries%20incorporating%20Dynamic%20Management%20Views%20(DMVs)%20that%20display%20information%20on%20some%20of%20the%20common%20SQL%20Server%20performance%20bottlenecks.)
* [Stumble](http://www.stumbleupon.com/submit?url=http://searchsqlserver.techtarget.com/tutorial/SQL-Server-queries-with-DMVs-for-examining-bottlenecks&title=SQL%20Server%20queries%20with%20DMVs%20for%20examining%20bottlenecks)
* [Delicious](http://del.icio.us/post?v=4&noui&jump=close&url=http://searchsqlserver.techtarget.com/tutorial/SQL-Server-queries-with-DMVs-for-examining-bottlenecks&title=SQL%20Server%20queries%20with%20DMVs%20for%20examining%20bottlenecks)

**Queries using DMVs**

Here are some queries incorporating DMVs that display information on some of the bottlenecks. Most of these DMV queries come from [Troubleshooting performance problems on SQL Server 2005](http://www.microsoft.com/technet/prodtechnol/sql/2005/tsprfprb.mspx):

 **Memory.** If you use the DMVs for memory, check how SQL Server internally allocates its memory. Run DBCC PROCCACHE to see how the total number of allocated buffers (num proc buffs) compares with the number used (num proc buffs used). A high-value percentage indicates poor use of procedure cache.

Run DBCC MEMORYSTATUS and observe the values for buffer distribution table. If the number of targeted pages decreases over time, it is likely that your SQL Server is experiencing external memory pressure. Compare the number of targeted pages against the stolen pages. If the number of stolen pages does not stabilize over time, the server may eventually get into internal physical memory pressure. Here is another SQL Server 2005 query that will expose these same counters:

SELECT \* FROM SYS.SYSPERFINFO WHERE  
OBJECT\_NAME='SQLSERVER:BUFFER MANAGER' AND  
(COUNTER\_NAME='TARGET PAGES' OR  
COUNTER\_NAME='TOTAL PAGES' OR  
COUNTER\_NAME='DATABASE PAGES' OR  
COUNTER\_NAME='STOLEN PAGES' OR  
COUNTER\_NAME='FREE PAGES')

Use the following DMV query to determine which SQL Server components are consuming the most amount of memory, and observe how this changes over time:

SELECT TYPE, SUM(MULTI\_PAGES\_KB) FROM  
SYS.DM\_OS\_MEMORY\_CLERKS WHERE  
MULTI\_PAGES\_KB != 0 GROUP BY TYPE

This query will show which SQL Server objects are consuming memory:

SELECT TYPE, PAGES\_ALLOCATED\_COUNT FROM  
SYS.DM\_OS\_MEMORY\_OBJECTS WHERE  
PAGE\_ALLOCATOR\_ADDRESS IN (SELECT TOP 10  
PAGE\_ALLOCATOR\_ADDRESS FROM  
SYS.DM\_OS\_MEMORY\_CLERKS ORDER BY  
MULTI\_PAGES\_KB DESC) ORDER BY  
PAGES\_ALLOCATED\_COUNT DESC

To get an idea of which individual processes are taking up memory, use the following query:

SELECT TOP 10 SESSION\_ID, LOGIN\_TIME, HOST\_NAME,  
PROGRAM\_NAME, LOGIN\_NAME, NT\_DOMAIN,   
NT\_USER\_NAME, STATUS, CPU\_TIME, MEMORY\_USAGE,   
TOTAL\_SCHEDULED\_TIME, TOTAL\_ELAPSED\_TIME,   
LAST\_REQUEST\_START\_TIME,  
LAST\_REQUEST\_END\_TIME, READS, WRITES,   
LOGICAL\_READS, TRANSACTION\_ISOLATION\_LEVEL,   
LOCK\_TIMEOUT, DEADLOCK\_PRIORITY, ROW\_COUNT,   
PREV\_ERROR FROM SYS.DM\_EXEC\_SESSIONS ORDER  
BY MEMORY\_USAGE DESC

To solve memory problems, see if SQL Server memory is correctly allocated. If so, ensure that the procedure cache has not been squeezed. Typical culprits eating up procedure cache are large stored procedures or large amounts of ad hoc SQL that are not parameterized. Then examine large memory consumers and objects to see if your applications need re-architecting. Processes consuming large amounts of memory should be redesigned to consume less memory.

 **Disk.**Processes that are disk intensive typically do not have the appropriate indexes or have poor execution plans. Here is a DMV query that lists the top 25 tables experiencing I/O waits.

SELECT TOP 25 DB\_NAME(D.DATABASE\_ID) AS  
DATABASE\_NAME,   
QUOTENAME(OBJECT\_SCHEMA\_NAME(D.OBJECT\_ID,   
D.DATABASE\_ID)) + N'.' +  
QUOTENAME(OBJECT\_NAME(D.OBJECT\_ID,  
D.DATABASE\_ID)) AS OBJECT\_NAME, D.DATABASE\_ID,   
D.OBJECT\_ID, D.PAGE\_IO\_LATCH\_WAIT\_COUNT,  
D.PAGE\_IO\_LATCH\_WAIT\_IN\_MS, D.RANGE\_SCANS,  
D.INDEX\_LOOKUPS FROM (SELECT DATABASE\_ID,   
OBJECT\_ID, ROW\_NUMBER() OVER (PARTITION BY  
DATABASE\_ID ORDER BY  
SUM(PAGE\_IO\_LATCH\_WAIT\_IN\_MS) DESC) AS  
ROW\_NUMBER, SUM(PAGE\_IO\_LATCH\_WAIT\_COUNT) AS  
PAGE\_IO\_LATCH\_WAIT\_COUNT,   
SUM(PAGE\_IO\_LATCH\_WAIT\_IN\_MS) AS  
PAGE\_IO\_LATCH\_WAIT\_IN\_MS,   
SUM(RANGE\_SCAN\_COUNT) AS RANGE\_SCANS,   
SUM(SINGLETON\_LOOKUP\_COUNT) AS  
INDEX\_LOOKUPS FROM  
SYS.DM\_DB\_INDEX\_OPERATIONAL\_STATS(NULL, NULL,   
NULL, NULL) WHERE PAGE\_IO\_LATCH\_WAIT\_COUNT > 0  
GROUP BY DATABASE\_ID, OBJECT\_ID ) AS D LEFT JOIN  
(SELECT DISTINCT DATABASE\_ID, OBJECT\_ID FROM  
SYS.DM\_DB\_MISSING\_INDEX\_DETAILS) AS MID ON  
MID.DATABASE\_ID = D.DATABASE\_ID AND  
MID.OBJECT\_ID = D.OBJECT\_ID WHERE  
D.ROW\_NUMBER>20 ORDER BY  
PAGE\_IO\_LATCH\_WAIT\_COUNT DESC

You can also generate a list of columns that should have indexes on them:

SELECT \* FROM SYS.DM\_DB\_MISSING\_INDEX\_GROUPS  
G JOIN SYS.DM\_DB\_MISSING\_INDEX\_GROUP\_STATS GS  
ON GS.GROUP\_HANDLE = G.INDEX\_GROUP\_HANDLE  
JOIN SYS.DM\_DB\_MISSING\_INDEX\_DETAILS D ON  
G.INDEX\_HANDLE = D.INDEX\_HANDLE

You can radically improve disk use with the correct index choices. Examine your execution plans carefully to improve disk use. You may need to change the RAID type or disk speeds to improve disk subsystem performance.

 **CPU.** One of the most frequent contributors to high CPU consumption is stored procedure recompilation. Here is a DMV that displays the list of the top 25 recompilations:

SELECT TOP 25 SQL\_TEXT.TEXT, SQL\_HANDLE,   
PLAN\_GENERATION\_NUM, EXECUTION\_COUNT, DB\_NAME(DBID) AS DbName,   
OBJECTID FROM SYS.DM\_EXEC\_QUERY\_STATS A  
CROSS APPLY SYS.DM\_EXEC\_SQL\_TEXT(SQL\_HANDLE)   
AS SQL\_TEXT WHERE PLAN\_GENERATION\_NUM >1  
ORDER BY PLAN\_GENERATION\_NUM DESC

This DMV lists the top CPU consumers:

SELECT TOP 50 SUM(QS.TOTAL\_WORKER\_TIME) AS  
TOTAL\_CPU\_TIME, SUM(QS.EXECUTION\_COUNT) AS  
TOTAL\_EXECUTION\_COUNT, COUNT(\*) AS  
NUMBER\_OF\_STATEMENTS, SQL\_TEXT.TEXT,   
QS.PLAN\_HANDLE FROM SYS.DM\_EXEC\_QUERY\_STATS  
QS CROSS APPLY  
SYS.DM\_EXEC\_SQL\_TEXT(SQL\_HANDLE) AS SQL\_TEXT  
GROUP BY SQL\_TEXT.TEXT,QS.PLAN\_HANDLE ORDER  
BY SUM(QS.TOTAL\_WORKER\_TIME) DESC

Other things that cause high CPU usage are bookmark lookups, bad parallelism and looping code.

**Final note**

When searching for bottlenecks, look for memory bottlenecks, then disk and finally CPU. Capture a baseline using System Monitor, SQL Profiler and DMVs to determine what is causing the bottleneck and if it can be solved by a hardware upgrade. Once you have a baseline, you are ready to start diagnosing the problem. In most cases, the solution will involve query tuning, query rewrites or re-architecting your solution. Many times, throwing hardware at the problem will not have the performance gains of simple index placement.