http://www.sqlservercentral.com/blogs/glennberry/archive/2010/03/09/easy-ways-to-detect-i\_2F00\_o-pressure-in-sql-server-2008.aspx

[Easy Ways to Detect I/O Pressure in SQL Server 2008](http://www.sqlservercentral.com/blogs/glennberry/archive/2010/03/09/easy-ways-to-detect-i_2F00_o-pressure-in-sql-server-2008.aspx)

It is pretty common for large, busy SQL Server instances to run into I/O bottlenecks. Even smaller, less busy systems often run into problems when they are not sized and configured correctly. As a database professional, you need to be able to detect when your SQL Server instances are experiencing poor performance due to I/O bottlenecks.

Quite often, you may need to convince someone from another part of the organization (such as a SAN engineer or server administrator) that your SQL Server instances are seeing decreased performance because of I/O issues.  It is also very possible that you have plenty of capacity, and that your I/O subsystem is configured correctly, but that you have some poorly written queries that are causing excessive disk I/O. Before you start talking to the rest of the organization, it is a good idea to know what the real issue is.

I have put together a small collection of queries (most of which are DMV queries) that should be very helpful in that effort. Even if you are not able to run PerfMon (because you don’t enough rights on the database server itself), you can still run these queries to get a pretty good idea what is going on from an I/O perspective.

This post is part of T-SQL Tuesday #004, which is hosted by Mike Walsh. You can see his invitation [here](http://www.straightpathsql.com/archives/2010/03/invitation-for-t-sql-tuesday-004-io/).

-- Some I/O Specific DMV Queries

-- March 2010

-- Always look at Avg Disk Sec/Read and Avg Disk Sec/Write

-- in PerfMon for each Physical Disk

-- Isolate top waits for server instance since last restart or statistics clear

-- Look for I/O specific waits at the top of the list

WITH Waits AS

(SELECT wait\_type, wait\_time\_ms / 1000. AS wait\_time\_s,

100. \* wait\_time\_ms / SUM(wait\_time\_ms) OVER() AS pct,

ROW\_NUMBER() OVER(ORDER BY wait\_time\_ms DESC) AS rn

FROM sys.dm\_os\_wait\_stats

WHERE wait\_type NOT IN ('CLR\_SEMAPHORE','LAZYWRITER\_SLEEP','RESOURCE\_QUEUE','SLEEP\_TASK'

,'SLEEP\_SYSTEMTASK','SQLTRACE\_BUFFER\_FLUSH','WAITFOR', 'LOGMGR\_QUEUE','CHECKPOINT\_QUEUE'

,'REQUEST\_FOR\_DEADLOCK\_SEARCH','XE\_TIMER\_EVENT','BROKER\_TO\_FLUSH','BROKER\_TASK\_STOP'

,'CLR\_MANUAL\_EVENT','CLR\_AUTO\_EVENT','DISPATCHER\_QUEUE\_SEMAPHORE'

, 'FT\_IFTS\_SCHEDULER\_IDLE\_WAIT','XE\_DISPATCHER\_WAIT', 'XE\_DISPATCHER\_JOIN'))

SELECT W1.wait\_type,

CAST(W1.wait\_time\_s AS DECIMAL(12, 2)) AS wait\_time\_s,

CAST(W1.pct AS DECIMAL(12, 2)) AS pct,

CAST(SUM(W2.pct) AS DECIMAL(12, 2)) AS running\_pct

FROM Waits AS W1

INNER JOIN Waits AS W2

ON W2.rn <= W1.rn

GROUP BY W1.rn, W1.wait\_type, W1.wait\_time\_s, W1.pct

HAVING SUM(W2.pct) - W1.pct < 95; -- percentage threshold

-- \*\*\* I/O Related Waits \*\*\*

-- ASYNC\_IO\_COMPLETION Occurs when a task is waiting for I/Os to finish

-- IO\_COMPLETION Occurs while waiting for I/O operations to complete.

-- This wait type generally represents non-data page I/Os.

-- Data page I/O completion waits appear

-- as PAGEIOLATCH\_\* waits

-- PAGEIOLATCH\_SH Occurs when a task is waiting on a latch for a buffer that

-- is in an I/O request. The latch request is in Shared mode.

-- Long waits may indicate problems with the disk subsystem.

-- PAGEIOLATCH\_EX Occurs when a task is waiting on a latch for a buffer that

-- is in an I/O request. The latch request is in Exclusive mode.

-- Long waits may indicate problems with the disk subsystem.

-- WRITELOG Occurs while waiting for a log flush to complete.

-- Common operations that cause log flushes

-- are checkpoints and transaction commits.

-- PAGELATCH\_EX Occurs when a task is waiting on a latch for a buffer that

-- is not in an I/O request. The latch request is in Exclusive mode.

-- BACKUPIO Occurs when a backup task is waiting for data, or is waiting for a

-- buffer in which to store data

-- Check for IO Bottlenecks (run multiple times, look for values above zero)

SELECT cpu\_id, pending\_disk\_io\_count

FROM sys.dm\_os\_schedulers

WHERE [status] = 'VISIBLE ONLINE'

ORDER BY cpu\_id;

-- Look at average for all schedulers (run multiple times, look for values above zero)

SELECT AVG(pending\_disk\_io\_count) AS [AvgPendingDiskIOCount]

FROM sys.dm\_os\_schedulers

WHERE [status] = 'VISIBLE ONLINE';

-- High Latch waits (SH and EX) indicates the I/O subsystem is too busy

SELECT wait\_type, waiting\_tasks\_count, wait\_time\_ms, signal\_wait\_time\_ms,

wait\_time\_ms - signal\_wait\_time\_ms AS [io\_wait\_time\_ms]

FROM sys.dm\_os\_wait\_stats

WHERE wait\_type IN('PAGEIOLATCH\_EX', 'PAGEIOLATCH\_SH', 'PAGEIOLATCH\_UP')

ORDER BY wait\_type;

-- File Names and Paths for TempDB and all user databases in instance

SELECT DB\_NAME([database\_id])AS [Database Name], [file\_id],

name, physical\_name, type\_desc

FROM sys.master\_files

WHERE [database\_id] > 4 AND [database\_id] <> 32767

OR [database\_id] = 2;

-- Things to look at:

-- Are data files and log files on different drives?

-- Is everything on C: drive?

-- Is TempDB on dedicated drives?

-- Are there multiple data files?

-- Which queries are causing the most IO operations (can take a few seconds)

SELECT TOP (20) total\_logical\_reads/execution\_count AS [avg\_logical\_reads],

total\_logical\_writes/execution\_count AS [avg\_logical\_writes],

total\_worker\_time/execution\_count AS [avg\_cpu\_cost], execution\_count,

total\_worker\_time, total\_logical\_reads, total\_logical\_writes,

(SELECT DB\_NAME(dbid) + ISNULL('..' + OBJECT\_NAME(objectid), '')

FROM sys.dm\_exec\_sql\_text([sql\_handle])) AS query\_database,

(SELECT SUBSTRING(est.[text], statement\_start\_offset/2 + 1,

(CASE WHEN statement\_end\_offset = -1

THEN LEN(CONVERT(nvarchar(max), est.[text])) \* 2

ELSE statement\_end\_offset

END - statement\_start\_offset

) / 2)

FROM sys.dm\_exec\_sql\_text(sql\_handle) AS est) AS query\_text,

last\_logical\_reads, min\_logical\_reads, max\_logical\_reads,

last\_logical\_writes, min\_logical\_writes, max\_logical\_writes,

total\_physical\_reads, last\_physical\_reads, min\_physical\_reads, max\_physical\_reads,

(total\_logical\_reads + (total\_logical\_writes \* 5))/execution\_count AS io\_weighting,

plan\_generation\_num, qp.query\_plan

FROM sys.dm\_exec\_query\_stats

OUTER APPLY sys.dm\_exec\_query\_plan([plan\_handle]) AS qp

WHERE [dbid] >= 5 AND (total\_worker\_time/execution\_count) > 100

ORDER BY io\_weighting DESC;

-- Calculates average stalls per read, per write,

-- and per total input/output for each database file.

SELECT DB\_NAME(database\_id) AS [Database Name], file\_id ,io\_stall\_read\_ms, num\_of\_reads,

CAST(io\_stall\_read\_ms/(1.0 + num\_of\_reads) AS NUMERIC(10,1))

AS [avg\_read\_stall\_ms],io\_stall\_write\_ms,

num\_of\_writes,CAST(io\_stall\_write\_ms/(1.0+num\_of\_writes) AS NUMERIC(10,1))

AS [avg\_write\_stall\_ms],io\_stall\_read\_ms + io\_stall\_write\_ms AS [io\_stalls],

num\_of\_reads + num\_of\_writes AS [total\_io], CAST((io\_stall\_read\_ms +

io\_stall\_write\_ms)/(1.0 + num\_of\_reads + num\_of\_writes) AS NUMERIC(10,1))

AS [avg\_io\_stall\_ms]

FROM sys.dm\_io\_virtual\_file\_stats(null,null)

ORDER BY avg\_io\_stall\_ms DESC;

-- Helps determine which database files on the entire instance have the most I/O bottlenecks

-- Analyze Database I/O, ranked by IO Stall%

WITH DBIO AS

(SELECT DB\_NAME(IVFS.database\_id) AS db,

CASE WHEN MF.type = 1 THEN 'log' ELSE 'data' END AS file\_type,

SUM(IVFS.num\_of\_bytes\_read + IVFS.num\_of\_bytes\_written) AS io,

SUM(IVFS.io\_stall) AS io\_stall

FROM sys.dm\_io\_virtual\_file\_stats(NULL, NULL) AS IVFS

INNER JOIN sys.master\_files AS MF

ON IVFS.database\_id = MF.database\_id

AND IVFS.file\_id = MF.file\_id

GROUP BY DB\_NAME(IVFS.database\_id), MF.[type])

SELECT db, file\_type,

CAST(1. \* io / (1024 \* 1024) AS DECIMAL(12, 2)) AS io\_mb,

CAST(io\_stall / 1000. AS DECIMAL(12, 2)) AS io\_stall\_s,

CAST(100. \* io\_stall / SUM(io\_stall) OVER()

AS DECIMAL(10, 2)) AS io\_stall\_pct,

ROW\_NUMBER() OVER(ORDER BY io\_stall DESC) AS rn

FROM DBIO

ORDER BY io\_stall DESC;

-- The queries below are database specific

USE yourdatabasename;

GO

-- Top Cached SPs By Total Physical Reads (SQL 2008).

-- Physical reads relate to disk I/O pressure

SELECT TOP(25) p.name AS [SP Name],qs.total\_physical\_reads AS [TotalPhysicalReads],

qs.total\_physical\_reads/qs.execution\_count AS [AvgPhysicalReads], qs.execution\_count,

ISNULL(qs.execution\_count/DATEDIFF(Second, qs.cached\_time, GETDATE()), 0) AS [Calls/Second],

qs.total\_elapsed\_time, qs.total\_elapsed\_time/qs.execution\_count

AS [avg\_elapsed\_time], qs.cached\_time

FROM sys.procedures AS p

INNER JOIN sys.dm\_exec\_procedure\_stats AS qs

ON p.[object\_id] = qs.[object\_id]

WHERE qs.database\_id = DB\_ID()

ORDER BY qs.total\_physical\_reads DESC;

-- Top Cached SPs By Total Logical Writes (SQL 2008).

-- Logical writes relate to both memory and disk I/O pressure

SELECT TOP(25) p.name AS [SP Name], qs.total\_logical\_writes AS [TotalLogicalWrites],

qs.total\_logical\_writes/qs.execution\_count AS [AvgLogicalWrites], qs.execution\_count,

ISNULL(qs.execution\_count/DATEDIFF(Second, qs.cached\_time, GETDATE()), 0) AS [Calls/Second],

qs.total\_elapsed\_time, qs.total\_elapsed\_time/qs.execution\_count AS [avg\_elapsed\_time],

qs.cached\_time

FROM sys.procedures AS p

INNER JOIN sys.dm\_exec\_procedure\_stats AS qs

ON p.[object\_id] = qs.[object\_id]

WHERE qs.database\_id = DB\_ID()

ORDER BY qs.total\_logical\_writes DESC;

-- Lists the top statements by average input/output usage for the current database

SELECT TOP(50) OBJECT\_NAME(qt.objectid) AS [SP Name],

(qs.total\_logical\_reads + qs.total\_logical\_writes) /qs.execution\_count AS [Avg IO],

SUBSTRING(qt.[text],qs.statement\_start\_offset/2,

(CASE

WHEN qs.statement\_end\_offset = -1

THEN LEN(CONVERT(nvarchar(max), qt.[text])) \* 2

ELSE qs.statement\_end\_offset

END - qs.statement\_start\_offset)/2) AS [Query Text]

FROM sys.dm\_exec\_query\_stats AS qs

CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) AS qt

WHERE qt.[dbid] = DB\_ID()

ORDER BY [Avg IO] DESC;

-- Helps you find the most expensive statements for I/O by SP

**Comments**