

Reviewing Windchill Operational Drive Performance

1 Jun 2010 11:24 AM | 0

There are a variety of hidden gems in SQL Server. One of these gems allows you to approximate the average operational latency for reads and writes by individual database files. These files may be used in the filegroups of Windchill or **tempdb** or the transaction logs for Windchill or **tempdb**. If you can see the operational latency, you can detect both poor assignments of drives and deterioration of drives. If you are in a virtual provisioning environment or SANs environment, it can often explain why performance changes (due to reassignments of LUN to different physical spindles). This is a good thing to do as part of a regular Windchill performance review.

RAID, SANs, and iSCSI drives can be replaced by their administrators, can have other databases placed on them, or can deteriorate with a possible pending failure. As the Windchill administrator, you may not be informed of these changes. It is good to detect the changes proactively instead of reactively (through user complaints about slowness or drive failure).

There are two types of files used in the database:

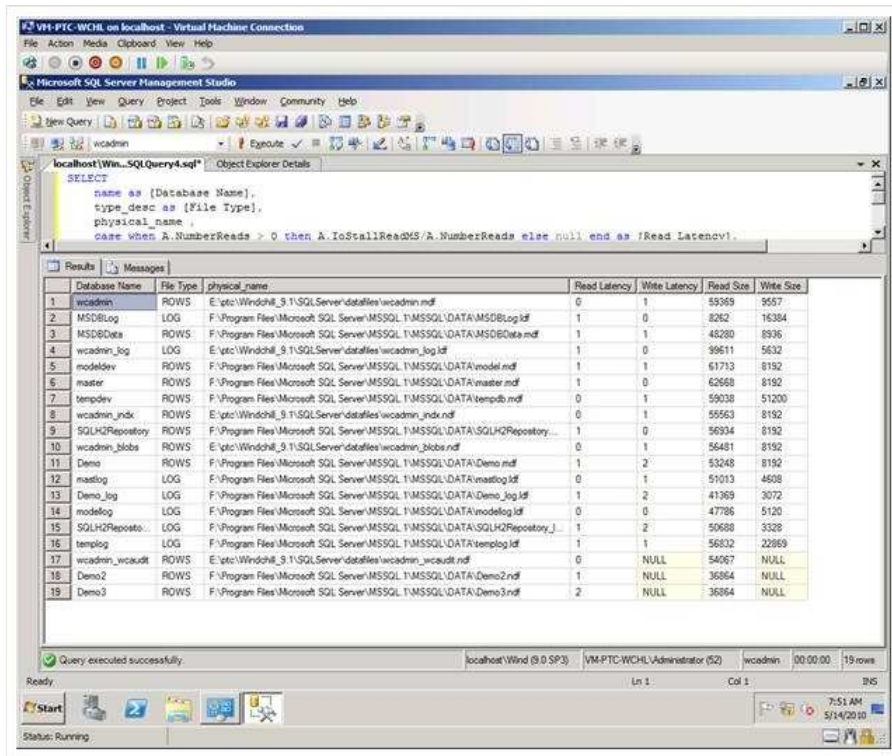
- ROWS: Random read/write occurs across the file.
- LOG: Sequential writes are appended to the file, and reads are done only in the event of a restoration.

Physical spindles perform differently for these two types of activity. For example, slow head movement or low spindle speed will not affect a LOG file as much as it will affect a ROWS file.

The following Transact-SQL statements return the type of the file and the actual location of the file (so that you know which drive it is on).

```
SELECT
    name as [Database Name],
    type_desc as [File Type],
    physical_name ,
    case when A.NumberReads > 0 then A.IoStallReadMS/A.NumberReads else null end as [Read Latency],
    case when A.NumberWrites > 0 then A.IoStallWriteMS/A.NumberWrites else null end as [Write Latency],
    case when A.NumberReads > 0 then A.BytesRead/A.NumberReads else null end as [Read Size],
    case when A.NumberWrites > 0 then A.BytesWritten/A.NumberWrites else null end as [Write Size]
FROM fn_virtualfilestats (NULL, NULL) A
JOIN sys.master_files C
    On database_id=DbId
    and file_id=FileId
    Order by A.NumberReads desc, A.NumberWrites desc
```

You may see results that resemble the following, in which the maximum average read and write latencies are 2 msec. NULL indicates no activity of a specific type has occurred.



On a different system, you may see results that resemble the following.

Database Name	Read Latency	Write Latency	Read Size	Write Size	Drive
tempdb	3	5	49603	57052	F
templog	3	3	55981	58404	F
wcadmin	325	40	74764	58616	K

In this case, it is clear that **tempdb** is located on the correct drive (the fastest drive). The write latency for **templog** is better than that for **tempdb**, which is to be expected because they are on the same drive and the log writes are sequential. The difference between the drives is almost shockingly bad. But, with a single drive for all the filegroups bouncing the drive head all over the place, it could be expected because of drive characteristics or deterioration. You cannot determine the exact problem, but you know that drive K is a problem. There are several reasonable approaches that you can take to resolve the problem:

- You can move the files to a different drive and see if there is an improvement. A different drive can be either:
 - A different existing logical drive.
 - A replacement for the physical drive that is being used for drive K.
- You can distribute the Windchill filegroups (BLOBS, INDX, AUDIT, and WCADMIN) to different drives. (For more information, see my earlier post.)

You can visualize how the hard drive acts when you add a single record to an existing table that has two indexes and one referential integrity (RI):

1. The drive head moves to the index record that is used for the RI and then finds it.
2. The drive head moves to the table record and then writes the records.
3. The drive head moves to index 1 and then writes a record.
4. The drive head moves to index 2 and then writes a record.
5. The drive head moves to the log and then writes a record.

You can also compute the average performance per logical drive. The following Transact-SQL statements show a correct calculation.

```
SELECT
    type_desc as [File Type],
    Left(physical_name,2) ,
    case when Sum(A.NumberReads) > 0 then Sum(A.IoStallReadMS)/Sum(A.NumberReads) else null end as [Read Latency],
    case when Sum(A.NumberWrites) > 0 then Sum(A.IoStallWriteMS)/Sum(A.NumberWrites) else null end as [Write Latency],
    case when Sum(A.NumberReads) > 0 then Sum(A.BytesRead)/Sum(A.NumberReads) else null end as [Read Size],
    case when Sum(A.NumberWrites) > 0 then Sum(A.BytesWritten)/Sum(A.NumberWrites) else null end as [Write Size]
FROM fn_virtualfilestats(NULL,NULL) A
JOIN sys.master_files C
    On database_id=DbId
    and file_id=FileId
group by Left(physical_name,2),type_desc
order by Left(physical_name,2)
```

Remember: Averaging averages is meaningless. You cannot copy the per filegroup averages into Microsoft Office Excel and then average them.

On this problematic system, you obtain the following information.

File Type	Read Latency	Write Latency	Read Size	Write Size	Drive
ROWS	5	0	54125	14311	B
LOG	0	1	27768	19852	C
ROWS	1	4	56368	89731	C
LOG	6	0	47753	14896	E
ROWS	7	102	32768	51200	G
ROWS	220	40	32768	17014	K
ROWS	0	0	32768	139426	L
ROWS	7	14	32768	166400	M

There are other databases running on this system, and some of these databases have a filegroup colocated with Windchill drives.

What becomes apparent is that drive G has a clear problem that may bottleneck performance. The times for drive K and drive M are also less than optimal. The issue of colocating other databases' filegroups on the same logical drive often arises when IT tries to trim storage costs or reduce server count. The other databases may be bouncing the physical drive's head around constantly, so your database is constantly waiting. If this is the case, the only solution is to move the filegroups to different logical drives.

The data for **fn_virtualfilestats** is cleared whenever you restart SQL Server. Unfortunately, it is not possible to reset the numbers with a running server or to tell SQL Server to keep the numbers.

The following factors can influence the numbers in the reports above:

- All drives should have write caching disabled. So, always double check the drives with the best performance.
- Failure to do partition alignment can increase the times by 30 percent or more. Unfortunately, the solution means repartitioning the drives.
- Other activities on the drive (especially in a RAID or SAN environment where disks may be shared) may increase the times.
- Failure to reorganize data, for example, by not rebuilding the index, can increase times.
- Beware of multiple logical drives on the same physical drives. The problem we are identifying is a physical drive performance/contention issue.
- Distributing related objects (tables, indexes and RI indexes) to different drives reduces contention.

How to Capture the Time of Day Performance

The time for reads and writes will likely increase with load. Load creates contention for drives and thus increases latency. You want to capture the performance during the day. Now I'll show you how to do that. You can capture the time of day performance by directly referencing the underlying table, **dm_io_virtual_file_stats**.

1. In a new database called **Monitoring**, create this table.

```
CREATE TABLE [dbo].[TrackingVirtualFileStats] (
  [database_id] [smallint] NOT NULL,
  [file_id] [smallint] NOT NULL,
  [sample_ms] [int] NOT NULL,
  [num_of_reads] [bigint] NOT NULL,
  [num_of_bytes_read] [bigint] NOT NULL,
  [io_stall_read_ms] [bigint] NOT NULL,
  [num_of_writes] [bigint] NOT NULL,
  [num_of_bytes_written] [bigint] NOT NULL,
  [io_stall_write_ms] [bigint] NOT NULL,
  [io_stall] [bigint] NOT NULL,
  [size_on_disk_bytes] [bigint] NOT NULL,
  RecordedAt DateTime Default (getDate())
)
```

2. Create this stored procedure.

```
Create Proc p_CaptureVirtualFileStats
AS
Declare @RecordDate DateTime
Set @RecordDate=GETDATE()
INSERT INTO [TrackingVirtualFileStats]
  ([database_id]
  ,[file_id]
  ,[sample_ms]
  ,[num_of_reads]
  ,[num_of_bytes_read]
  ,[io_stall_read_ms]
  ,[num_of_writes]
  ,[num_of_bytes_written]
  ,[io_stall_write_ms]
  ,[io_stall]
  ,[size_on_disk_bytes]
  ,[RecordedAt])
  Select [database_id]
  ,[file_id]
  ,[sample_ms]
  ,[num_of_reads]
  ,[num_of_bytes_read]
  ,[io_stall_read_ms]
  ,[num_of_writes]
  ,[num_of_bytes_written]
  ,[io_stall_write_ms]
  ,[io_stall]
  ,[size_on_disk_bytes]
  ,@RecordDate from sys.dm_io_virtual_file_stats(-1,-1)
GO
```

3. Schedule an hourly call of the **p_CaptureVirtualFileStats** stored procedure during the workday to capture the data.

The following stored procedures provide an analysis. I suggest that you do an analysis whenever problems are reported (reactive analysis) and at the end of the month (proactive analysis).

To list the available times, run the following code.

```
Create proc p_ListVirtualFileStatsTimes as
Select Distinct RecordedAt
From [TrackingVirtualFileStats] order by RecordedAt
```

To obtain the performance between two of the times that you select from the list that is returned above, run the following code.

```
Create proc p_DeltaVirtualFileStatsTimes @FromTime DateTime, @ToTime DateTime
as
Select case when A.NumberReads > 0 then A.IoStallReadMS/A.NumberReads else null end as [Read Latency],
       case when A.NumberWrites > 0 then A.IoStallWriteMS/A.NumberWrites else null end as [Write Latency],
```

```

    case when A.NumberReads > 0 then A.BytesRead/A.NumberReads else null end as [Read Size],
    case when A.NumberWrites > 0 then A.BytesWritten/A.NumberWrites else null end as [Write Size],
    A.*
From
(Select
    DatabaseName=db_Name(FromData.[database_id])
    ,FileGroupPath=physical_name
    ,NumberReads=ToData.[num_of_reads]-FromData.[num_of_reads]
    ,BytesRead=ToData.[num_of_bytes_read]-FromData.[num_of_bytes_read]
    ,IOStallReadMs=ToData.[io_stall_read_ms]-FromData.[io_stall_read_ms]
    ,NumberWrites=ToData.[num_of_writes]-FromData.[num_of_writes]
    ,BytesWritten=ToData.[num_of_bytes_written]-FromData.[num_of_bytes_written]
    ,IOStallWriteMs=ToData.[io_stall_write_ms]-FromData.[io_stall_write_ms]
    ,DiskSizeDelta=ToData.[size_on_disk_bytes]-FromData.[size_on_disk_bytes]
From [TrackingVirtualFileStats] FromData
JOIN [TrackingVirtualFileStats] ToData
On FromData.database_id=ToData.database_id
And FromData.File_id=ToData.File_id
JOIN sys.master_files C
    On FromData.database_id=C.database_id
    and C.file_id=FromData.File_Id
Where FromData.[RecordedAt]=@FromTime
And ToData.[RecordedAt]=@ToTime
) A

```

The **p_DeltaVirtualFileStatsTimes** procedure produces the same output that is shown in the first screenshot. The data that is shown is only for the specific time period.

Bottom Line

So what is the bottom-line benefit to you? You can detect where there is a physical problem behind a Windchill performance problem. Typically, you'll see one of three problems:

- Slow hardware is being used.
- The hardware is deteriorating. For example, the drive has to repeatedly read a track to get the data. For more information, see [SMART](#).
- Other databases are colocated.

In my experience, checking for these physical issues is typically the first thing to do before you head down the road of database tuning. Why? It's fast to do and eliminates or confirms one class of problems quickly.

Ken Lassenen is part of the original team that created Dr. GUI of MSDN and specializes in new and resurrected commercial product architecture. He developed architecture for several Microsoft websites, including the original [MSDN](#) site and the current [Microsoft Partner Network](#) site. He's equally at home with SQL Server, XHTML, Section 508 accessibility standards, globalization, Security Content Automation Protocol (SCAP) security, C#, and ASP.NET server controls. When he is not having fun with technology, he enjoys taking lunch-break hikes in the North Cascades.

