It’s that time again, T-SQL Tuesday is here! This time Pat Wright ([blog](http://voiceofthedba.wordpress.com/)|[twitter](http://twitter.com/#!/SqlAsylum)) is hosting and has put forth automating tasks using ether T-SQL or Powershell. I LOVE automating stuff. As a production DBA in some very large shops you can’t do your job unless you make your servers work for you. I’ve been using T-SQL and \*GASP\* xp\_cmdshell, OSQL and file shares to gather stats and push configurations to servers for decades. Log before fancy things like C# and Powershell existed. These days I use a variety of home grown tools but doing things with just T-SQL can be just as powerful. I’m going to leverage this post to start a series on a pure T-SQL implementation of configuration management, data gathering and utility procedure deployment.

**Where Is Your Management Database?**

Every DBA should have two things, a utility, or management database on every server and a central repository where all the locally collected data in the management database is pulled back to. What surprises people when I talk about this methodology is I don’t always advocate using a licensed copy of SQL Server. By that I mean that SQL Express works just great as a central repository server. I usually put this on a virtual machine, Hyper-V or what ever flavor of virtual environment your company supports. This allows you to do things like enable CLR and xp\_cmdshell on a server that technically is non-production and keep your security risk much lower. Every server that is deployed in my shop gets a local management database. From the central repository I can push out everything I need to manage a server and keep all my servers up to date as I make improvements or bug fixes to the management code. That’s all I’m really going to say about that in this post though I just wanted to give you an idea of just how deep the rabbit hole can go.

**DMV’s give you the information, you have to use it.**

Since SQL Sever 2005 Microsoft let of of the black box mentality and started providing crazy useful information via Dynamic Management Views. Virtual file statistics though have been around for quite a while. They got a touch up in the DMV but the basic information was available in SQL Server 2000 via function call.  The DMV I’m after is sys.dm\_io\_virtual\_file\_stats. It has a ton of information in it. It’s main problem though is it is an aggregation over time and doesn’t really tell you what is different from yesterday to today. To get around that we have to build our own sampling routine.

**The VirtualFileStats Table**

We create a table in our local management database to collect the daily numbers. I try to keep things simple.

[view source](http://sqlserverio.com/#viewSource)



[print](http://sqlserverio.com/#printSource)[?](http://sqlserverio.com/#about)

|  |  |  |
| --- | --- | --- |
| 01 | create table dbo.VirtualFileStats ( | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | RecordID int IDENTITY (1, 1) NOT NULL , | |
| 04 | ServerName varchar(255) not null, |

|  |  |
| --- | --- |
| 05 | DBID int NOT NULL , |
| 06 | FileID int NOT NULL , | |

|  |  |
| --- | --- |
| 07 | Reads bigint NULL , |
| 08 | ReadsFromStart bigint NULL , | |

|  |  |
| --- | --- |
| 09 | Writes bigint NULL , |
| 10 | WritesFromStart bigint NULL , | |

|  |  |
| --- | --- |
| 11 | BytesRead bigint NULL , |
| 12 | BytesReadFromStart bigint NULL , | |

|  |  |
| --- | --- |
| 13 | BytesWritten bigint NULL , |
| 14 | BytesWrittenFromStart bigint NULL , | |

|  |  |
| --- | --- |
| 15 | IostallInMilliseconds bigint NULL , |
| 16 | IostallInMillisecondsFromStart bigint NULL , | |

|  |  |
| --- | --- |
| 17 | IostallReadsInMilliseconds bigint NULL , |
| 18 | IostallReadsInMillisecondsFromStart bigint NULL , | |

|  |  |
| --- | --- |
| 19 | IostallWritesInMilliseconds bigint NULL , |
| 20 | IostallWritesInMillisecondsFromStart bigint NULL , | |

|  |  |
| --- | --- |
| 21 | RecordedDateTime datetime NULL , |
| 22 | IntervalInMilliseconds bigint NULL , | |

|  |  |  |
| --- | --- | --- |
| 23 | FirstMeasureFromStart bit NULL | |
| 24 | ) |

This is what we need to gather, and later analyze the data. Since we are managing our samples we have to know when the sampling started and what the first sample is. FirstMeasureFromStart lets us know that it is the first base measurements the rest of the samples will delta off of.

**GatherVirtualFileStats Stored Procedure**

Next we need a stored procedure to do the sampling. One thing you will notice is the procedure executes continuously with a WAIT FOR DELAY built into it so you can get finer grained than the 1 minute limitation of the SQL Agent. Sometimes, I will do one off sampling for a short period, say 30 minutes at a 30 second interval but most often I just let it run and set the sample rate at 1 minute or larger depending on how busy the system is.

[view source](http://sqlserverio.com/#viewSource)



[print](http://sqlserverio.com/#printSource)[?](http://sqlserverio.com/#about)

|  |  |
| --- | --- |
| 001 | IF EXISTS (SELECT \* |
| 002 | FROM   dbo.sysobjects | |

|  |  |  |
| --- | --- | --- |
| 003 | WHERE  id = Object\_id(N'[dbo].[GatherVirtualFileStats]') | |
| 004 | AND Objectproperty(id, N'IsProcedure') = 1) |

|  |  |  |
| --- | --- | --- |
| 005 | DROP PROCEDURE [dbo].[GatherVirtualFileStats] | |
| 006 |  |

|  |  |  |
| --- | --- | --- |
| 007 | GO | |
| 008 |  |

|  |  |  |
| --- | --- | --- |
| 009 | --------------------------------------------------------------------------------------- | |
| 010 | --  GatherVirtualFileStats |

|  |  |
| --- | --- |
| 011 |  |
| 012 | --  by: Wesley D. Brown | |

|  |  |
| --- | --- |
| 013 | --  date: 02/08/2011 |
| 014 | --  mod:  00/00/0000 |

|  |  |
| --- | --- |
| 015 |  |
| 016 | --  description: | |

|  |  |  |
| --- | --- | --- |
| 017 | --  This stored procedure is used to sample sys.dm\_io\_virtual\_file\_stats to track | |
| 018 | --  performance at a database file level. This is useful for finding |

|  |  |  |
| --- | --- | --- |
| 019 | --  hotspots on SAN's or under performing IO systems. | |
| 020 |  |

|  |  |
| --- | --- |
| 021 | --  parameters: |
| 022 | --    @Duration  = '01:00:00' How long to run before exiting | |

|  |  |  |
| --- | --- | --- |
| 023 | --   @IntervalInSeconds = 120 Number of seconds between samples | |
| 024 | --@DB            = -1 DB\_ID to monitor, -1 for all |

|  |  |  |
| --- | --- | --- |
| 025 | --@DBFile        = -1 File\_ID of file to monitor, -1 for all | |
| 026 | --  usage: |

|  |  |
| --- | --- |
| 027 | --      DECLARE @RC         INT, |
| 028 | --          @StartTime  DATETIME, | |

|  |  |  |
| --- | --- | --- |
| 029 | --          @databaseID INT | |
| 030 |  |

|  |  |  |
| --- | --- | --- |
| 031 | --  SELECT @StartTime = Getdate(), | |
| 032 | --         @databaseID = Db\_id() |

|  |  |
| --- | --- |
| 033 |  |
| 034 | --  EXEC @RC = Gathervirtualfilestats | |

|  |  |  |
| --- | --- | --- |
| 035 | --    '00:45:30', | |
| 036 | --    30, |

|  |  |  |
| --- | --- | --- |
| 037 | --    10, | |
| 038 | --    -1 |

|  |  |
| --- | --- |
| 039 |  |
| 040 | --  SELECT \* | |

|  |  |  |
| --- | --- | --- |
| 041 | --  FROM   dbo.VirtualFileStats | |
| 042 | --  WHERE  DBID = 10 |

|  |  |  |
| --- | --- | --- |
| 043 | --  ORDER  BY RecordID | |
| 044 |  |

|  |  |
| --- | --- |
| 045 | --  platforms: |
| 046 | --  SQL Server 2005 | |

|  |  |
| --- | --- |
| 047 | --  SQL Server 2008 |
| 048 | --  SQL Server 2008 R2 | |

|  |  |
| --- | --- |
| 049 | --  tested: |
| 050 | --  SQL Server 2005 SP2 | |

|  |  |
| --- | --- |
| 051 | --  SQL Server 2008 R2 |
| 052 | --------------------------------------------------------------------------------------- | |

|  |  |
| --- | --- |
| 053 | --  \*\*\* change log      \*\*\* |
| 054 | --  \*\*\* end change log  \*\*\* |

|  |  |  |
| --- | --- | --- |
| 055 | ------------------------------------------------------------------------------------- | |
| 056 | CREATE PROC dbo.Gathervirtualfilestats @Duration    DATETIME = '01:00:00', |

|  |  |  |
| --- | --- | --- |
| 057 | @IntervalInSeconds INT = 120, | |
| 058 | @DB          INT = -1, |

|  |  |  |
| --- | --- | --- |
| 059 | @DBFile      INT = -1 | |
| 060 | AS |

|  |  |
| --- | --- |
| 061 | DECLARE @StopTime                 DATETIME, |
| 062 | @LastRecordedDateTime     DATETIME, |

|  |  |  |
| --- | --- | --- |
| 063 | @CurrentDateTime          DATETIME, | |
| 064 | @ErrorNumber              INT, |

|  |  |
| --- | --- |
| 065 | @NumberOfRows             INT, |
| 066 | @ErrorMessageText         NVARCHAR(4000), | |

|  |  |  |
| --- | --- | --- |
| 067 | @CurrentServerName        VARCHAR(255), | |
| 068 | @DifferenceInMilliSeconds BIGINT |

|  |  |
| --- | --- |
| 069 |  |
| 070 | SELECT @CurrentServerName = CAST(Serverproperty('servername') AS VARCHAR(255)) | |

|  |  |
| --- | --- |
| 071 |  |
| 072 | SET @DifferenceInMilliSeconds = Datediff(ms, CONVERT(DATETIME, '00:00:00', 8), @Duration) | |

|  |  |
| --- | --- |
| 073 |  |
| 074 | SELECT @StopTime = Dateadd(ms, @DifferenceInMilliSeconds, Getdate()) | |

|  |  |
| --- | --- |
| 075 |  |
| 076 | WHILE Getdate() <= @StopTime | |

|  |  |
| --- | --- |
| 077 | BEGIN |
| 078 | SELECT @LastRecordedDateTime = @CurrentDateTime | |

|  |  |
| --- | --- |
| 079 |  |
| 080 | SELECT @CurrentDateTime = Getdate() | |

|  |  |
| --- | --- |
| 081 |  |
| 082 | INSERT INTO dbo.VirtualFileStats | |

|  |  |  |
| --- | --- | --- |
| 083 | (ServerName, | |
| 084 | DBID, |

|  |  |  |
| --- | --- | --- |
| 085 | FileID, | |
| 086 | Reads, |

|  |  |  |
| --- | --- | --- |
| 087 | ReadsFromStart, | |
| 088 | Writes, |

|  |  |  |
| --- | --- | --- |
| 089 | WritesFromStart, | |
| 090 | BytesRead, |

|  |  |  |
| --- | --- | --- |
| 091 | BytesReadFromStart, | |
| 092 | BytesWritten, |

|  |  |
| --- | --- |
| 093 | BytesWrittenFromStart, |
| 094 | IostallInMilliseconds, |

|  |  |  |
| --- | --- | --- |
| 095 | IostallInMillisecondsFromStart, | |
| 096 | IostallReadsInMilliseconds, |

|  |  |  |
| --- | --- | --- |
| 097 | IostallReadsInMillisecondsFromStart, | |
| 098 | IostallWritesInMilliseconds, |

|  |  |  |
| --- | --- | --- |
| 099 | IostallWritesInMillisecondsFromStart, | |
| 100 | RecordedDateTime, |

|  |  |  |
| --- | --- | --- |
| 101 | IntervalinMilliseconds, | |
| 102 | FirstMeasureFromStart) |

|  |  |
| --- | --- |
| 103 | SELECT @CurrentServerName, |
| 104 | vfs.database\_id -- Database ID | |

|  |  |
| --- | --- |
| 105 | , |
| 106 | vfs.[file\_id] -- File ID | |

|  |  |
| --- | --- |
| 107 | , |
| 108 | vfs.num\_of\_reads - dbaf.ReadsFromStart                            AS Reads, | |

|  |  |  |
| --- | --- | --- |
| 109 | vfs.num\_of\_reads                                                  AS ReadsFromStart, | |
| 110 | | vfs.num\_of\_writes - dbaf.WritesFromStart                          AS Writes, |

|  |  |  |
| --- | --- | --- |
| 111 | vfs.num\_of\_writes                                                 AS WritesFromStart, | |
| 112 | | vfs.num\_of\_bytes\_read - dbaf.BytesReadFromStart                   AS BytesRead, |

|  |  |  |
| --- | --- | --- |
| 113 | vfs.num\_of\_bytes\_read                                             AS BytesReadFromStart, | |
| 114 | | vfs.num\_of\_bytes\_written - dbaf.BytesWrittenFromStart             AS BytesWritten, |

|  |  |  |
| --- | --- | --- |
| 115 | vfs.num\_of\_bytes\_written                                          AS BytesWrittenFromStart, | |
| 116 | | vfs.io\_stall - dbaf.IostallInMillisecondsFromStart                AS IostallInMilliseconds, |

|  |  |  |
| --- | --- | --- |
| 117 | vfs.io\_stall                                                      AS IostallInMillisecondsFromStart, | |
| 118 | | vfs.io\_stall\_read\_ms - dbaf.IostallReadsInMillisecondsFromStart   AS IostallReadsInMilliseconds, |

|  |  |  |
| --- | --- | --- |
| 119 | vfs.io\_stall\_read\_ms                                              AS IostallReadsInMillisecondsFromStart, | |
| 120 | | vfs.io\_stall\_write\_ms - dbaf.IostallWritesInMillisecondsFromStart AS IostallWritesInMilliseconds, |

|  |  |  |  |
| --- | --- | --- | --- |
| 121 | vfs.io\_stall\_write\_ms                                             AS IostallWritesInMillisecondsFromStart, | | |
| 122 | | @CurrentDateTime, |

|  |  |
| --- | --- |
| 123 | CASE |
| 124 | WHEN @LastRecordedDateTime IS NULL THEN NULL | |

|  |  |  |
| --- | --- | --- |
| 125 | | ELSE Datediff(ms, dbaf.RecordedDateTime, @CurrentDateTime) |
| 126 | END                                                               AS IntervalInMilliseconds, | |

|  |  |
| --- | --- |
| 127 | CASE |
| 128 | WHEN @LastRecordedDateTime IS NULL THEN 1 | |

|  |  |  |
| --- | --- | --- |
| 129 | | ELSE 0 |
| 130 | END                                                               AS FirstMeasureFromStart | | |

|  |  |  |
| --- | --- | --- |
| 131 | FROM   sys.Dm\_io\_virtual\_file\_stats(@DB, @DBFile) vfs | |
| 132 | LEFT OUTER JOIN VirtualFileStats dbaf |

|  |  |
| --- | --- |
| 133 | ON vfs.database\_id = dbaf.dbid |
| 134 | AND vfs.[file\_id] = dbaf.fileid | |

|  |  |
| --- | --- |
| 135 | WHERE  ( @LastRecordedDateTime IS NULL |
| 136 | OR dbaf.RecordedDateTime = @LastRecordedDateTime ) | |

|  |  |
| --- | --- |
| 137 |  |
| 138 | SELECT @ErrorNumber = @@ERROR, | |

|  |  |  |
| --- | --- | --- |
| 139 | @NumberOfRows = @@ROWCOUNT | |
| 140 |  |

|  |  |  |
| --- | --- | --- |
| 141 | IF @ErrorNumber != 0 | |
| 142 | BEGIN |

|  |  |  |
| --- | --- | --- |
| 143 | SET @ErrorMessageText = 'Error ' + CONVERT(VARCHAR(10), @ErrorNumber) + ' failed to insert file stats data!' | |
| 144 |  |

|  |  |  |
| --- | --- | --- |
| 145 | RAISERROR (@ErrorMessageText, | |
| 146 | 16, |

|  |  |  |
| --- | --- | --- |
| 147 | 1) WITH LOG | |
| 148 |  |

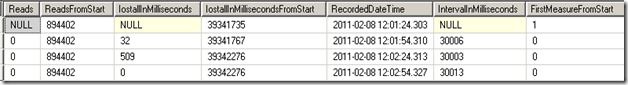
|  |  |  |
| --- | --- | --- |
| 149 | RETURN @ErrorNumber | |
| 150 | END |

|  |  |
| --- | --- |
| 151 |  |
| 152 | WAITFOR DELAY @IntervalInSeconds | |

|  |  |
| --- | --- |
| 153 | END |

**I Have Data, Now What?**

This is where the fun begins! If you just query the table the data doesn’t make much sense.

[](http://sqlserverio.files.wordpress.com/2011/02/image.png)

We need to do some simple math and get the data into a metric that is meaningful.

[view source](http://sqlserverio.com/#viewSource)



[print](http://sqlserverio.com/#printSource)[?](http://sqlserverio.com/#about)

|  |  |
| --- | --- |
| 01 | SELECT TOP 10 Db\_name(dbid)                                          AS 'databasename', |
| 02 | File\_name(fileid)                                      AS 'filename', |

|  |  |
| --- | --- |
| 03 | Reads / ( IntervalInMilliSeconds / 1000 )              AS 'readspersecond', |
| 04 | Writes / ( IntervalInMilliSeconds / 1000 )             AS 'writespersecond', |

|  |  |  |
| --- | --- | --- |
| 05 | ( Reads + Writes ) / ( IntervalInMilliSeconds / 1000 ) AS 'iopersecond', | |
| 06 | CASE |

|  |  |
| --- | --- |
| 07 | WHEN ( Reads / ( IntervalInMilliSeconds / 1000 ) ) > 0 |
| 08 | AND IostallReadsInMilliseconds > 0 THEN IostallReadsInMilliseconds / Reads | |

|  |  |
| --- | --- |
| 09 | ELSE 0 |
| 10 | END                                                    AS 'iolatencyreads', | |

|  |  |
| --- | --- |
| 11 | CASE |
| 12 | WHEN ( Reads / ( IntervalInMilliSeconds / 1000 ) ) > 0 | |

|  |  |  |
| --- | --- | --- |
| 13 | AND IostallWritesInMilliseconds > 0 THEN IostallWritesInMilliseconds / Writes | |
| 14 | ELSE 0 |

|  |  |  |
| --- | --- | --- |
| 15 | END                                                    AS 'iolatencywrites', | |
| 16 | CASE |

|  |  |
| --- | --- |
| 17 | WHEN ( ( Reads + Writes ) / ( IntervalInMilliSeconds / 1000 ) > 0 |
| 18 | AND IostallInMilliseconds > 0 ) THEN IostallInMilliseconds / ( Reads + Writes ) |

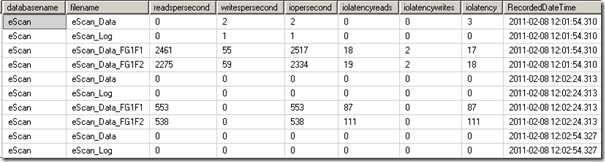
|  |  |
| --- | --- |
| 19 | ELSE 0 |
| 20 | END                                                    AS 'iolatency', | |

|  |  |
| --- | --- |
| 21 | RecordedDateTime |
| 22 | FROM   management.dbo.VirtualFileStats | |

|  |  |
| --- | --- |
| 23 | WHERE  DBID = 10 |
| 24 | AND FirstMeasureFromStart = 0 | |

|  |  |
| --- | --- |
| 25 | ORDER  BY RecordID |

This gives us reads, writes and io latency per second results.

[](http://sqlserverio.files.wordpress.com/2011/02/image1.png)

Now we are cooking! We can now see that on this database we are seeing some spikes in latency, the number of milliseconds it takes to complete a single IO request, and may warrant investigation. As a general rule of thumb if I see IO latency above 20 milliseconds consistently I start looking deeper into the IO system to see what is wrong. A single modern hard disk is capable of 130 random IO’s a second. Another thing to consider is how many databases are on the same disk. I will pull the database files together with the volume they are on to get a true aggregate number of IO’s and latencies. You may find that a single database is dominating the disks and causing other databases to slow down even when the number of IO’s for those databases is small. On a SAN being able to get to the file level can help you locate the physical disks for that LUN and help your SAN administrators look at a very specific spot instead of the overall health of the SAN and actually fix your performance issues.

Lastly, I run a nightly job that performs these aggregations for me and moves them into a table that I keep long term so I can see performance over time. This is a great way to see if you are getting near your IO capacity and if the steps you have done correct the issues.

**So, That’s it?**

Not by a long shot. Measuring IO latencies is just one piece of a much larger puzzle for troubleshooting performance problems. It is a valuable one though, and keeping this information over time is just priceless.