SQL Server Disk Stall Collector

Setup and Usage Guide

SQL Server records ongoing disk latency metrics in the form of 'disk stall' values, as collected by SQL Server on an ongoing basis. The challenge is that these stall metrics are cumulative from the last time that the SQL Server service was started, instead of an average value on a sample window. Therefore, an ongoing collector can be set up to store these cumulative values on a fixed time interval, and then calculate the latency averages on a small granular window of time. These metrics provide a deeper look into the latency values, and can be used to analyze the ongoing latency state of the server's disks.

First, either create a new database or select a DBA utility database. This example uses a new database called CollectorTemp.

```
USE [CollectorTemp]
GO

CREATE TABLE [dbo].[DiskStall](
	[ID] [bigint] IDENTITY(1,1) NOT NULL,
	[ServerName] [nvarchar](128) NULL,
	[InstanceName] [nvarchar](128) NULL,
	[DatabaseID] [smallint] NOT NULL,
	[DatabaseName] [sysname] NOT NULL,
	[PhysicalFileName] [nvarchar](260) NOT NULL,
	[DatabaseFileID] [int] NOT NULL,
	[AvgReadStallms] [numeric](10, 1) NULL,
	[AvgWriteStallms] [numeric](10, 1) NULL,
	[IOStallReadms] [bigint] NULL,
	[NumOfReads] [bigint] NULL,
	[IOStallWritems] [bigint] NULL,
	[NumOfWrites] [bigint] NULL,
	[TotalNumOfBytesRead] [bigint] NULL,
	[TotalNumOfBytesWritten] [bigint] NULL,
	[CollectionDateTime] [datetime] NOT NULL,
 CONSTRAINT [PK_DiskStall] PRIMARY KEY CLUSTERED
  ( [ID] ASC )
) ON [PRIMARY]
```

Next, create a stored procedure in this database that performs our metric collection and previous sample differential calculation.

```
CREATE PROCEDURE dbo.usp_DiskStallCollection AS

SET NOCOUNT ON;

BEGIN TRY

-- Calculates average stalls per read, per write, and
-- per total input/output for each database file.
;WITH PreviousSample_CTE( ServerName, InstanceName, DatabaseID, DatabaseFileID,
	IOStallReadms, NumOfReads, IOStallWritems, NumOfWrites) AS (
		SELECT
			ServerName, InstanceName, DatabaseID, DatabaseFileID,
			IOStallReadms, NumOfReads, IOStallWritems, NumOfWrites
		FROM dbo.DiskStall WHERE CollectionDateTime =
		     (SELECT MAX(CollectionDateTime) FROM dbo.DiskStall)
)
INSERT INTO dbo.DiskStall
	([ServerName]
    ,[InstanceName]
```

```
    ,[DatabaseID]
    ,[DatabaseName]
    ,[PhysicalFileName]
    ,[DatabaseFileID]
    ,[AvgReadStallms]
    ,[AvgWriteStallms]
    ,[IOStallReadms]
    ,[NumOfReads]
    ,[IOStallWritems]
    ,[NumOfWrites]
    ,[TotalNumOfBytesRead]
    ,[TotalNumOfBytesWritten]
    ,[CollectionDateTime])
SELECT
@@SERVERNAME as ServerName, @@SERVICENAME as InstanceName,
fs.database_id as DatabaseID, d.name as DatabaseName,
mf.physical_name as PhysicalFileName, mf.file_id as DatabaseFileID,
CAST((io_stall_read_ms - (isnull(cte.IOStallReadms,0.0)))/
       (1.0 + (num_of_reads - isnull(cte.NumOfReads,0.0))) AS NUMERIC(10,1)) AS [AvgReadStallms],
CAST((io_stall_write_ms - (isnull(cte.IOStallWritems,0.0)))/
       (1.0 + (num_of_writes-isnull(cte.NumOfWrites,0.0))) AS NUMERIC(10,1)) AS [AvgWriteStallms],
io_stall_read_ms as IOStallReadms,
num_of_reads as NumOfReads,
io_stall_write_ms as IOStallWritems,
num_of_writes as NumOfWrites,
num_of_bytes_read as TotalNumOfBytesRead,
num_of_bytes_written as TotalNumOfBytesWritten,
GETDATE() as CollectionDateTime
FROM sys.dm_io_virtual_file_stats(null,null) AS fs
INNER JOIN sys.master_files AS mf ON fs.database_id = mf.database_id
     AND fs.[file_id] = mf.[file_id]
inner join sys.databases as d on d.database_id = fs.database_id
left join PreviousSample_CTE cte on cte.ServerName = ServerName
       and cte.InstanceName = InstanceName
       and cte.DatabaseID = fs.database_id
       and cte.DatabaseFileID = mf.file_id
ORDER BY
fs.database_id asc OPTION (RECOMPILE);

END TRY
BEGIN CATCH
       THROW
END CATCH;
```

Create a SQL Server Agent scheduled job that executes this stored procedure on an ongoing basis. A recommended value is every 15 seconds if a latency problem is suspected, or one minute if no problem is currently under review.

To view the raw values per database data and log file, execute the following query.

```
select * from dbo.DiskStall order by CollectionDateTime, ServerName, DatabaseName
```

This query dumps the raw data, along with the calculated values for disk read and write latency over that sample period. However, analyzing these values over a longer window of time are necessary for determining the running state health of

the environment. Performing a percentile analysis provides us a much more granular and significant view of the performance of the environment. To perform a percentile analysis, this SQL Server 2012 and above query will determine the running percentile metrics for the sample window. Adjust the time window being queried to match your specific requirements.

```
select distinct
       DatabaseID, ServerName,
       DatabaseName, PhysicalFileName, DatabaseFileID,
       round(percentile_cont(0) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms00,
       round(percentile_cont(0.25) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms25,
       round(percentile_cont(0.5) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms50,
       round(percentile_cont(0.75) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms75,
       round(percentile_cont(0.9) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms90,
       round(percentile_cont(0.95) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms95,
       round(percentile_cont(0.96) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms96,
       round(percentile_cont(0.97) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms97,
       round(percentile_cont(0.98) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms98,
       round(percentile_cont(0.99) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms99,
       round(percentile_cont(0.999) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms99d9,
       round(percentile_cont(1) within group (order by AvgReadStallms)
              over (partition by PhysicalFileName),2) as ReadStallms100,
       round(percentile_cont(0) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms00,
       round(percentile_cont(0.25) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms25,
       round(percentile_cont(0.5) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms50,
       round(percentile_cont(0.75) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms75,
       round(percentile_cont(0.9) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms90,
       round(percentile_cont(0.95) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms95,
       round(percentile_cont(0.96) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms96,
       round(percentile_cont(0.97) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms97,
       round(percentile_cont(0.98) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms98,
       round(percentile_cont(0.99) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms99,
       round(percentile_cont(0.999) within group (order by AvgWriteStallms)
              over (partition by PhysicalFileName),2) as WriteStallms99d9,
       round(percentile_cont(1) within group (order by AvgWriteStallms)
```

```
              over (partition by PhysicalFileName),2) as WriteStallms100
from
       dbo.DiskStall
order by
       DatabaseName, DatabaseFileID
```