**Analyze and Improve SQL Server Backup Schedule**

By: [Aakash Patel](https://www.mssqltips.com/sqlserverauthor/340/aakash-patel/)   |   Updated: 2019-11-22

FROM: <https://www.mssqltips.com/sqlservertip/6221/analyze-and-improve-sql-server-backup-schedule/>

**Problem**

A centralized backup server to support SQL Server can sometimes become a resource bottleneck in the backup stream if you have many servers writing database backups as the same time. It usually happens when you have a large SQL Server database inventory and all of your database servers have [similar backup schedules](https://www.mssqltips.com/sqlservertip/2521/standardize-sql-server-backups-and-maintain-existing-job-schedules/). In this tip, we will look into SQL Server backup schedules and how you can use this to reduce backup server contention and possibly making [SQL Server backups](https://www.mssqltips.com/sqlservertutorial/1/sql-server-backup-options-and-commands-tutorial/) faster by simply rescheduling backup jobs.

**Solution**

It is considered a best practice to have a separate backup server from where your database files reside to avoid a single point of failure. When you have a large inventory of SQL Server database servers and all of them are backing up at or around the same time, it may overwhelm the backup server and become sluggish. In turn, backups will run for a longer period and will likely impact transaction log backups and other processing on the server during that time window. If transaction log backups take longer, the transaction log can grow and become large and possibly fill up the drive on the server and may not comply with your RPO agreement either.

If you don’t have a very strict maintenance window, you can stagger the backup operations and spread them out across weekdays and times. You can use the below script to audit the full backup schedule. The script is created in the context to run it on [CMS](https://www.mssqltips.com/sql-server-tip-category/140/central-management-servers/) to get a nice overview of backup schedule for the entire server inventory. The query result should give you a clear understanding of how to stagger the backup jobs to reduce overlapping time windows.

Potentially, spreading out backup operations will make backups faster across the board. In our environment, we were able reduce backup duration by 30-50% on servers with large and lots of database(s) by simply avoiding an overlapping schedule. You can use the same script below to monitor your [SQL Server differential backups](https://www.mssqltips.com/sqlservertip/1233/differential-database-backups-for-sql-server/) schedule with a minor update of changing the code below from this **bs.type to 'd'** to this **bs.type to 'i'**.

The below script will also inform you about databases with uncompressed backups. To use compressed backups, make sure to turn on the server level configuration ‘[backup compression default](https://www.mssqltips.com/sqlservertip/1964/sql-server-2008-backup-compression/)’ if it was missed during server setup. Backup compression has been around for since SQL Server 2008 and is available in both Standard and Enterprise editions since SQL Server 2008 R2.

Here is the script to run.

SELECT

bkup.compressed,

COUNT(\*) AS total\_dbs,

bkup.[weekday],

DATEDIFF(MINUTE, MIN(bkup.backup\_start\_date), MAX(bkup.backup\_finish\_date)) AS duration\_minutes,

AVG(speed\_mb\_sec) avg\_mb\_sec,

MIN(bkup.backup\_start\_date) AS backup\_began,

MAX(bkup.backup\_finish\_date) AS backup\_finished

FROM (SELECT

DATENAME(WEEKDAY, bs.backup\_start\_date) AS [weekday],

bs.backup\_start\_date,

bs.backup\_finish\_date,

speed\_mb\_sec = (bs.compressed\_backup\_size / 1048576.0) /

CASE

WHEN DATEDIFF(SECOND, bs.backup\_start\_date, bs.backup\_finish\_date) > 0 THEN DATEDIFF(SECOND, bs.backup\_start\_date, bs.backup\_finish\_date)

ELSE 1

END,

CASE

WHEN (bs.backup\_size % bs.compressed\_backup\_size) > 0 THEN '1'

ELSE '0'

END [compressed],

RANK() OVER (PARTITION BY bs.database\_name ORDER BY bs.backup\_start\_date DESC) AS rank

FROM msdb..backupset bs (NOLOCK)

WHERE 1 = 1

AND bs.type = 'd' --full backups

AND bs.is\_copy\_only = 0

AND bs.database\_name NOT IN ('master', 'msdb', 'model') --excludes server with no user dbs

AND bs.backup\_start\_date > DATEADD(DAY, -7, GETDATE()) --backups in last one week

--and datename(weekday,bs.backup\_start\_date) = 'Monday' -- filter by day

) bkup

WHERE bkup.rank = 1 -- latest full backup

GROUP BY bkup.compressed,

bkup.[weekday];

Here is sample output.  This shows the last time full backups were run on the server.  I shows how many backups are compressed, number of backups, overall time for the backups and when the backups started and finished.

database backup information

**Next Steps**

* Try splitting backups into multiple files to make it faster
  + [Backup to multiple files for faster and smaller SQL Server files](https://www.mssqltips.com/sqlservertip/935/backup-to-multiple-files-for-faster-and-smaller-sql-server-files/)
  + [SQL Server Script to Automatically Split Database Backups into Multiple Backup Files](https://www.mssqltips.com/sqlservertip/5668/sql-server-script-to-automatically-split-database-backups-into-multiple-backup-files/)
* Explore native BACKUP options such as MAXTRANSFERSIZE, BUFFERCOUNT, etc.
  + [Options to Improve SQL Server Backup Performance](https://www.mssqltips.com/sqlservertip/2539/options-to-improve-sql-server-backup-performance/)

Last Updated: 2019-11-22