**[Automated Backup Tuning](https://sirsql.net/2012/12/13/20121212automated-backup-tuning/)**

FROM: <https://sirsql.net/2012/12/13/20121212automated-backup-tuning/>

Tuning your backups is a wonderful thing. You could easily reduce the time it takes to backup your databases by 50% of more just by changing a few settings, and yet so few people do it. I asked myself the question why and came up with two answers.

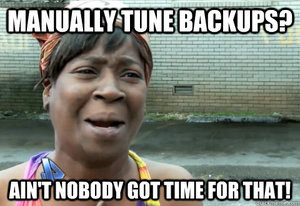
1. People do not know how to tune their backups.
2. It takes too long to run through the tuning process.

How to tune your backups

I’m not going to go over methods for tuning your backups in this post. There are people who have done a far better job at both [explaining the adjustments you can make](http://sqlcat.com/sqlcat/b/technicalnotes/archive/2009/02/16/tuning-backup-compression-part-2.aspx) and that have [created videos](http://midnightdba.itbookworm.com/Video/Watch?VideoId=290) to help you understand and perform the process yourself.

My biggest concern was directed at the level of effort required to test all the possible permutations of files, maxtransfersize and buffercount values, after all, who has time to figure all of that out and then capture the data to look at the relative performance characteristics of each one?

I decided that the best way to do this was to create a nice little test harness which would run through all those tests without manual intervention, and then figure out a way to get the output from all of those backups into some kind of meaningful graph for that instant visual on performance differences.

[](http://67.225.208.62/~sirsql/content/wp-content/uploads/2012/12/NobodyGotTimeForThat.jpg)  
No manual backup tuning for this DBA

**The backup testing script**

Step one in the automated tuning is a [SQL script](https://sirsql.net/content/PostScripts/Backup-Test-Harness.sql) I created which accepts a few input variables:

* @DatabaseName – name of the database you want to use for testing
* @MaxBackupFiles – maximum number of files you want to write to at a time
* @BackupFolder – destination folder for the backups
* @WithCompression – whether or not to use SQL Server backup compression
* @CreateJob – Whether or not to create a SQL Agent job to run the tests
* @JobLogFileName – file path and name for a log for the job
* @OutputToScreen – outputs the backup commands to the screen

When executed the script is designed to created SQL Agent job which you can then execute at a time of your choosing. The job will run through a series of backups for a database (at minimum 40) and capture the information in the job log file.

Be warned, performing this many backups test, with so many settings, can and will impact your server performance at some point. Be aware of this. Do not cause issues for your users when performing the tests, they will not like you for doing so.

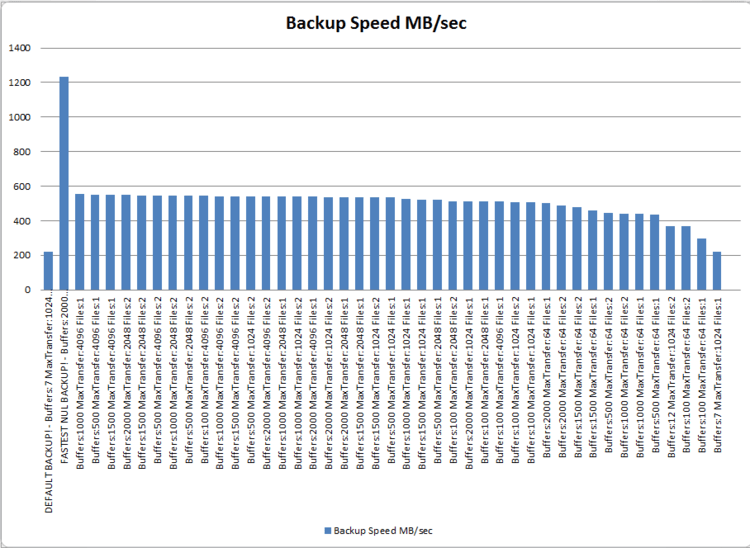
**Visualizing the results**

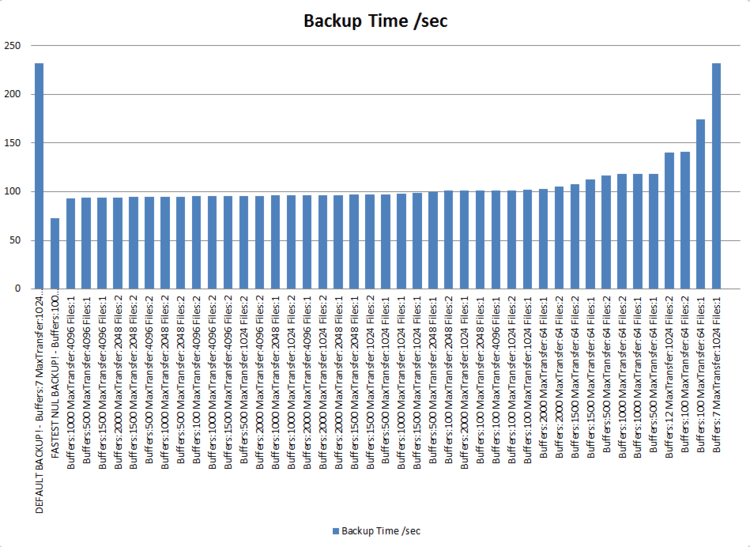
Once the backup testing job is complete a job log file is left behind. This log file contains all of the information for each of the backups performed including the:

* number of files backed up to
* maxtranfersize setting used
* buffercount setting used
* speed of the backup in MB/sec
* time the backup took to complete in seconds

Getting that information usable is done using a[PowerShell script](https://sirsql.net/content/PostScripts/Parse-Backup-Perf-Tests.ps1). A blog post by Jonathan Kehayis described his method for [parsing SQLIO output to excel charts](http://sqlblog.com/blogs/jonathan_kehayias/archive/2010/05/25/parsing-sqlio-output-to-excel-charts-using-regex-in-powershell.aspx). Based on this script, the update version parses the backup log file that has been created, imports the information into excel and creates a couple of charts.

These charts show the speed (Figure 1) and backup time (Figure 2) for the default backup options, the absolute fastest backup to NUL, and the all of the other options that were tested.

[](http://67.225.208.62/~sirsql/content/wp-content/uploads/2012/12/Backup-Speed.png)  
Figure 1. Backup speed results for 100GB database backed up across 10Gb network to CIFS share

[](http://67.225.208.62/~sirsql/content/wp-content/uploads/2012/12/Backup-Time.png)  
Figure 2. Backup time results for 100GB database backed up across 10Gb network to CIFS share

The graphs provide a nice visual representation of the backup performance which helps to quickly identify that there are improvements that can be made.

The default SQL backup runs at just over 200MB/sec and takes about 235 seconds to complete, but we know that we can perform that same backup to a NUL device in about 75 seconds at a rate of over 1200MB/sec.

Just looking through the rest of the data it is clear to see that a lot of the values used make a negligible change to the performance of the backups, so we can work to make an informed decision on what the best file, maxtransfersize and buffercount settings should be.

Hopefully this is something that you can find useful. Please give it a try and send me feedback on your testing efforts.

Downloads: [Backup Test Harness.sql](https://sirsql.net/content/PostScripts/Backup-Test-Harness.sql), [Parse Backup Perf Tests.ps1](https://sirsql.net/content/PostScripts/Parse-Backup-Perf-Tests.ps1)

**2 thoughts on “Automated Backup Tuning”**

1. https://0.gravatar.com/avatar/cdcef32266ed2ebfaa708df194b31ab6?s=32&d=identicon&r=G[**Jimmy May @AspiringGeek**](http://sqlblog.com/blogs/jimmy_may) says:

[June 15, 2015 at 6:50 pm](https://sirsql.net/2012/12/13/20121212automated-backup-tuning/#comment-58)

I’ve heard this post referenced in several presentations–including of course @TheresaIserman’s superb presentation. I was thinking about you today during @PaulRandal’s discussion of striping backups. Nice job, Nic, thank you.

[Like](https://sirsql.net/2012/12/13/20121212automated-backup-tuning/?like_comment=58&_wpnonce=22781f8bd2)Like

[Reply](https://sirsql.net/2012/12/13/20121212automated-backup-tuning/?replytocom=58#respond)

1. https://1.gravatar.com/avatar/7ac1901b16cb51c5471c050900973413?s=32&d=identicon&r=G**Martin Guth** says:

[December 30, 2015 at 9:00 am](https://sirsql.net/2012/12/13/20121212automated-backup-tuning/#comment-59)

Great Stuff!  
I had a bit of a rough start with the powershell script. First time it was executed I got a blank excel sheet. After a bit of debugging it turned out, that Excel on my system creates a new workbook ( Command $WorkBook = $Excel.WorkBooks.Add();) just with one sheet. Thus the subsequent commands $WorkBook.WorkSheets.Item(3).Delete()  
$WorkBook.WorkSheets.Item(2).Delete()  
did fail. It was not that obvious to me from the start as the catch block is empty and it just seemed the script has done nothing. Switching to line by line execution after having removed the catch block I spotted the error and fixed it. Maybe this is helpful for some other readers as well. My advice: If you run into problems with the result try removing the try, catch first in order to get a decent error message. Powershell practicioners would easily be able to print something like a stack trace inside the catch block…however my powershell skills are not that elaborate.

Thanks again for the great work. I’ve messed around a bit with VBA in Excel years ago and it just seems to me that you can do a lot of the same stuff within powershell. That’s quite cool. However I have to admit that meanwhile as a DW/BI professional I try to avoid this kind of excel programming and rather resort to a data entry system on top of a database.