# Optimizing backup speed

The backup process can be separated into three distinct stages:

- Stage 1: SQL Server backup engine reads the data and log files.This incurs disk Input/Output (I/O) reads.
- Stage 2: SQL Backup Pro compresses, and optionally encrypts, the data.This uses CPU cycles.
- Stage 3: SQL Backup Pro writes the resulting compressed data files to disk.This incurs disk I/O writes.

To optimize the speed of your backup, you optimize each of these stages in turn:

1. Use the SQL Backup Pro command line or extended stored procedure to run a backup using the `NOCOMPRESSWRITE` argument (see The BACKUP command).
   When you use `NOCOMPRESSWRITE`, SQL Backup Pro simulates a backup process without compression, and no backup files are created. This simulates Stage 1.
   The throughput for a backup process is shown on the SQL Server report. When you use the `NOCOMPESSWRITE` argument, this shows the maximum possible backup throughput attainable on your system.
   If you are using a multi-processor system, test the effect of using multiple threads. You are recommended to start with one thread fewer than the number of processors. For example, if you are using four processors, start with three threads.
   The limiting factor for Stage 1 is usually the speed at which your disk can read the data from the disks.
2. Use SQL Backup Pro to run a backup using the NOWRITE argument (see The BACKUP command).
   When you use the NOWRITE argument, SQL Backup Pro simulates a backup process using the specified compression level and no backup files are created. This simulates Stage 1 and Stage 2.
   You can change the compression level to see the effect that the different levels have on the backup speed.
   To improve backup throughput, use multiple threads until you achieve a throughput close to that you achieved with the `NOCOMPRESSWRITE` argum ent.
3. Run the backup process to completion, using the optimum number of threads found at Stage 2.
   To optimize Stage 3, store the backup data on a different set of disks from the disks used to store the data and log files. Depending upon the type of disk controllers, you have, you may need to reduce the number of backup devices so that you do not overload the disk I/O writes. Your aim is to balance the number of backup devices with the number of disks you can back up, to achieve a throughput close to that achieved at Stage 2.
   The Current Disk Queue Length performance counter tells you when you have reached the maximum capacity of your disks. Generally, the number should not exceed twice the number of drives on your disk array. If you achieve this, your backup process has been optimized.

   

   **BUFFERCOUNT**

   SQL Server Backup Pro also supports BUFFERCOUNT. This can also be used to significantly increase performance. It is recommended to modify this parameter on its interaction with other parameters such as FILECOUNT is non-trivial.

   For information about using the BUFFERCOUNT option, see the Incorrect BufferCount data transfer option can lead to OOM condition blog.

## Example

1. A backup was run using `NOCOMPRESSWRITE`. SQL Server reported a throughput of 103 MB/sec. The number of threads was increased to two. Throughput remained at 103 MB/sec. Therefore, the limiting factor for this setup is the disk read speed, at 103 MB/sec.
2. The same backup was run using `NOWRITE`, with compression level 1.
   SQL Server reported a throughput of 60 MB/sec. The number of threads was increased to two, and throughput increased to 99 MB/sec. The number of threads was then increased to three, and throughput increased to 103 MB/sec. This is the maximum disk read speed found at Stage 1. The process was then repeated for compression level 2. With one thread, throughput was 80 MB/sec; with two and three threads, throughput was 103 MB/sec.
   Therefore, optimum backup speed is achieved using compression level 2 with two threads.
3. The same backup was then run to completion (without the `NOCOMRESSWRITE` or `NOWRITE` arguments) using compression level 2 with two threads.
   Throughput was 77 MB/sec. The number of threads was increased to 3, and SQL Server reported a throughput drop to 71 MB/sec. Therefore, two threads is optimum choice for this particular backup on this system. The *Current Disk Read Queue* performance counter averaged 13 for a 4 disk array.