Introduction

Due to an increased awareness about data mining, text mining and big data applications across all domains the value of data has been realized and is resulting in data sets with a large number of variables and increased observation size. Often it takes a lot of time to process these datasets which can have an impact on delivery timelines. When there is limited permanent storage space, storing such large datasets may cause serious problems. Best way to handle some of these constraints is by making a large dataset smaller, by reducing the number of observations and/or variables or by reducing the size of the variables, without losing valuable information. In this paper we will see how a SAS data set can be compressed using the COMPRESS= option and also some techniques to make this option more effective.

The process of reducing the number of bytes required to represent each observation is known as Compression. Some of the advantages of compressing a dataset include but are not limited to, a compressed file reduces the storage requirements and also reduces the number of I/O operations necessary to read or write the data during processing. In a compressed file, the deleted observation space can be reused using the COMPRESS= option, whereas in an uncompressed data set the deleted observation space is never reused.

To create a compressed data set we use the COMPRESS= output data set option or system option.

Syntax: SAS-data-set(Compress=N0|YES) CHAR(BINARY)

COMPRESS=NO|YES|CHAR|BINARY

Two data sets are being used here, one contains character data and the other binary data.

Compressing Numeric Data

The option COMPRESS=BINARY uses the Ross Data Compression (RDC) algorithm, which combines run-length encoding and sliding window compression. This option is more effective with observations greater than a thousand bytes in length and can be very effective with numeric data and character data that contain patterns rather than simple repetitions. It takes significantly more CPU time to uncompress than COMPRESS=YES|CHAR.

options fullsetter:

PROC IMPORT OUT=
  WORK.bindata(compress=binary)
  DATAFILE= "H:\compress.csv"
  DBMS=CSV REPLACE;
  GETNAMES=YES;
  DATAROW=2;
  RUN;

MEASURING EFFICIENCY

The following are the statistics for data compressed using COMPRESS=BINARY option

NOTE: 45 records were read from the input file "H:\compress.csv".
The original record length was 675.
The compressed record length was 675.
The data set WORK.BINDATA has 45 observations and 20 variables.
NOTE: Compressing data set WORK.BINDATA decreased size by 81.87 percent.
Compressed to 19 pages, un compressed would require 25 pages.

The size of the data set decreased by 81.82 percent after compressing and also the number of pages required by compressed data set decreased to 10 from 55.

Results:

Text Data:

Before Compressing:

CPU time: 1.29 seconds
CPU time: 0.34 seconds
Memory: 1277.4M

After Compressing:

CPU time: 0.18 seconds
Compressed file size: 0.12 seconds
Memory: 1277.4M

From the above performance statistics we can see that the processing time for compressed data set is 1.1 seconds which is 0.19 seconds less compared to the uncompressed version of the file. There is also significant decrease in memory due to data set compression.

Numerical Data:

Before Compressing:

CPU time: 1.29 seconds
CPU time: 0.34 seconds
Memory: 1277.4M

After Compressing:

CPU time: 0.18 seconds
Compressed file size: 0.12 seconds
Memory: 1277.4M

The processing time for compressed data set is 1 second where as for uncompressed is 1.2. our sample data contains only 425 observations but in many practical applications the data set contains millions of observations so the difference in processing time will be very significant.

Making Compress= Option more effective

Some of you might be wondering why SAS does not store the data as small as possible, automatically compress them during data step processing. There are several reasons why we don’t favor the "blind" use of this SAS system option in all situations. One of the main reasons is increase in CPU time. Additional CPU resources are required to both create a compressed data set and to apply SAS tasks to the compressed data set. The SAS system must uncompress the data set every time it applies a data step or procedure step to it.

Following are few techniques which can be used along with the compress= option to make the process of decreasing the size of a large data set more effective.

1. LENGTH STATEMENT: The default length of variables is 8 bytes. The length statement can be used to declare the byte length of variables in the Program Data Vector. More observations can be placed in each page of a dataset memory by appropriately adjusting the byte lengths of variables, resulting in a smaller data set and also reducing the number of input/output operations required to move the data set in and out of the CPU.

2. COMPRESS= option: When you create a compressed file, you can specify COMPRESS= (as a data set option or a system option) in order to track and reuse space. With COMPRESS=YES, new observations are inserted in space which becomes freed when other observations are updated or deleted, which results in optimal usage of the space. Note that when using COMPRESS=es and KEEP=es system options, the observations cannot be addressed using observation numbers. This is because the COMPRESS= option has higher precedence over KEEP=es option and they are mutually exclusive.

3. OPTIMIZING CPU PERFORMANCE: The major drawback of compressing data set is increase in CPU time, by setting the CGOPTIMIZE= option to 0 CPU time can be saved when processing large data sets. So Optimizing CPU performance while compressing, using techniques like these can improve the overall performance.

CONCLUSION

Data Set Compression reduces the size of the data sets only if the maximum size of the observation is more than the 12-byte (32-bit systems) or 24-byte (64-bit system) overhead introduced by compression. A compressed data set requires less number of pages which results in less number of I/O operations and increased performance. Nevertheless, it should be noted that there is an increase in CPU time for processing the compressed file, so if you have limited CPU resources then compressing a data set might not be a good option.