Paper 258-2013

Data Set Compression using COMPRESS=

ABSTRACT

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= system option and also some techniques to make this option more effective.

INTRODUCTION

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 REUSE= 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=NO| YES| CHAR| BINARY)

OPTIONS COMPRESS=NO|YES|CHAR|BINARY

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

COMPRESSING TEXT DATA

The option COMPRESS=YES|CHAR is effective with character data that contains repeated characters such as blanks. It uses the run-length encoding (RLE) compression algorithm, which compresses repeating consecutive bytes such as trailing blanks or repeated zeroes.

```
options fullstimer;

PROC IMPORT OUT= WORK.charcompress(compress=yes)

            DATAFILE= "H:\compress.csv"

            DBMS=CSV REPLACE;

     GETNAMES=YES;

     DATAROW=2;

RUN;
```

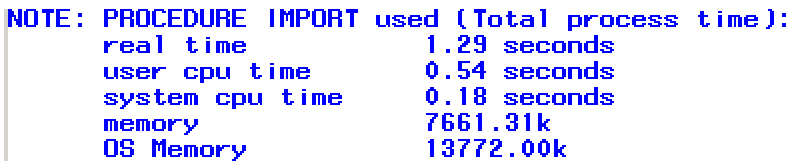
MEASURING EFFICIENCY

The following are the statistics for compressed data set

```
NOTE: The data set WORK.CHARCOMPRESS has 2514 observations and 99 variables.
NOTE: Compressing data set WORK.CHARCOMPRESS decreased size by 36.11 percent.
      Compressed is 230 pages; un-compressed would require 360 pages.
```

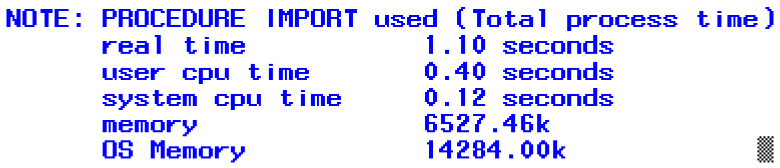
The size of the data set decreased by 36.11 percent and the number of pages decreased are 130, which significantly decreases the number of I/O operations. There will be increase in CPU time but the savings in I/O operations greatly outweigh the increase in CPU time.

Before Compressing:



```
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           1.29 seconds
      user cpu time       0.54 seconds
      system cpu time     0.18 seconds
      memory              7661.31k
      OS Memory           13772.00k
```

After Compressing:



```
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           1.10 seconds
      user cpu time       0.40 seconds
      system cpu time     0.12 seconds
      memory              6527.46k
      OS Memory           14284.00k
```

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.

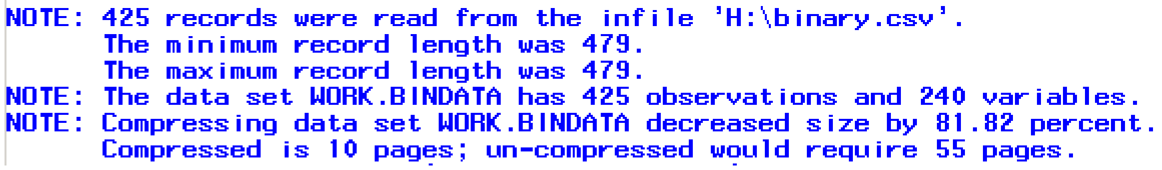
COMPRESSING NUMERIC DATA

The option COMPRESS=BINARY uses the Ross Data Compression (RDC) algorithm, which combines run-length encoding and sliding window compression. This option is more efficient 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 fullstimer;

PROC IMPORT OUT= WORK.bindata(compress=binary)

            DATAFILE= "H:\binary.csv"

            DBMS=CSV REPLACE;

     GETNAMES=YES;

     DATAROW=2;

RUN;
```

MEASURING EFFICIENCY

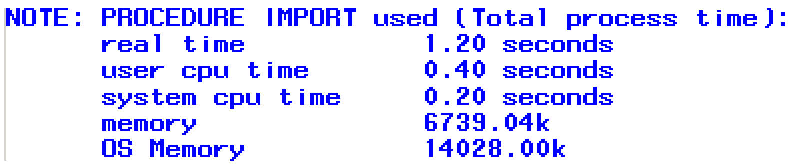
The following are the statistics for data compressed using COMPRESS=BINARY option



```
NOTE: 425 records were read from the infile 'H:\binary.csv'.
      The minimum record length was 479.
      The maximum record length was 479.
NOTE: The data set WORK.BINDATA has 425 observations and 240 variables.
NOTE: Compressing data set WORK.BINDATA decreased size by 81.82 percent.
      Compressed is 10 pages; un-compressed would require 55 pages.
```

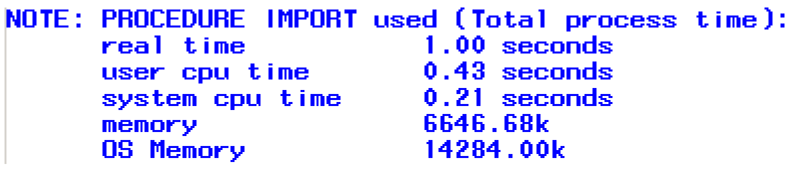
The size of the data set decreased by 81.82 percent after compressing and also the number of pages required by compressed data set decreased to 10 from 55.

Before Compression:



```
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           1.20 seconds
      user cpu time       0.40 seconds
      system cpu time     0.20 seconds
      memory              6739.04k
      OS Memory           14028.00k
```

After Compression



```
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           1.00 seconds
      user cpu time       0.43 seconds
      system cpu time     0.21 seconds
      memory              6646.68k
      OS Memory           14284.00k
```

The processing time for compressed data set is 1 second where as for uncompressed is 1.2, our sample data set 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, for the sake of making all data sets 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 numeric variables is 8 bytes. The Length statement can be used to declare the byte lengths 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. REUSE= OPTION: When you create a compressed file, you can also specify REUSE=YES (as a data set option or a system option) in order to track and reuse space. With REUSE=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=yes and reuse=yes system options, the observations cannot be addressed using observation numbers. This is because the REUSE= option has higher precedence over POINTOBS= 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.

REFERENCES

"HTTP://SUPPORT.SAS.COM/DOCUMENTATION/CDL/EN/LRDICT/64316/HTML/DEFAULT/VIEWER.HTM#A000202890.HTM"

"HTTP://SUPPORT.SAS.COM/DOCUMENTATION/CDL/EN/LRDICT/64316/HTML/DEFAULT/VIEWER.HTM#A000202894.HTM"

ANDREW H. KARP "INDEXING AND COMPRESSING SAS®DATA SETS: HOW, WHY AND WHY NOT" SUGI28 PAPER 3-28

"HTTP://SUPPORT.SAS.COM/DOCUMENTATION/CDL/EN/LRCON/62955/HTML/DEFAULT/VIEWER.HTM#A001091115.HTM"

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Srinivas Busi Reddy, Email: srinivas.busi_reddy@okstate.edu
Srinivas Busi Reddy is Master's student in Management Information Systems at Oklahoma State University. He is an ADVANCED SAS® 9 certified Programmer.

Srikar Rayabaram, Email: rayabar@okstate.edu
Srikar Rayabaram is Master's student in Management Information Systems at Oklahoma State University. He is a BASE SAS® 9 and a certified SAS® predictive modeler using Enterprise Miner 6. He will be receiving his SAS® and OSU Data Mining Certificate in December 2012.

Musthan Meeran Mohideen, Email: musthan@okstate.edu
Musthan Meeran Mohideen is Master's student in Management Information Systems at Oklahoma State University. He is a BASE SAS® 9 and a certified SAS® predictive modeler using Enterprise Miner 6.