

43: To COMPRESS or Not, to COMPRESS or ZIP

David B. Horvath, CCP, MS



To COMPRESS or Not

Copyright © 2017- 2021, David B. Horvath, CCP, All Rights Reserved

The Author can be contacted at:

504 Longbotham Drive, Aston PA 19014-2502, USA

Phone: 1-610-859-8826

Email: dhorvath@cobs.com

Web: <http://www.cobs.com/>

LinkedIn: <https://www.linkedin.com/in/dbhorvath/>

All trademarks and servicemarks are the property of their respective owners.



Introductions

- My Background
- SAS Compress Basics
- SAS Compress Examples
- Operating System/Tool Compression
- Compression Comparison
- Taking Advantage of Parallelism – Piping



Abstract

- SAS supports both basic (Character) and advanced (binary) compression
- Operating systems and tools support additional compression.
- This session reviews the processing tradeoffs between uncompressed and SAS-compressed datasets as well as dealing with operating system compressed files and datasets.
- Is it better to process an uncompressed dataset or use SAS compression? What are the factors that influence the decision to compress (or not)? What are the considerations around applying operating system based compression (for example, Winzip or UNIX zip or GNU gzip) to regular files and SAS datasets? What are the tradeoffs? How can files in those formats be best processed in SAS?



My Background

- David is an IT Professional who has worked with various platforms since the 1980's with a variety of development and analysis tools.
- He has previously presented at PhilaSUG, SESUG, and SGF and has presented workshops and seminars in Australia, France, the US, Canada, and Oxford England (about the British Author Nevil Shute) for various organizations.
- He holds an undergraduate degree in Computer and Information Sciences from Temple University and a Masters in Organizational Dynamics from UPENN. He achieved the Certified Computing Professional designation with honors.
- Most of his career has been in consulting (although recently he has been in-house) in the Philadelphia PA area. He is currently in Data Analytics "Engineering" at a Regional Bank.
- He has several books to his credit (none SAS related) and has instructed as an Adjunct Instructor at various Colleges and Universities.



SAS Compress Basics

- Initially added with Version 6
 - And only removed extra spaces from strings
- Significant improvements with Version 8
 - Char or Yes: remove repeating blanks, characters, or numbers
 - Binary: Char plus Compress Numeric Variables
- Silent improvements with Version 9:
 - Much faster (less I/O) now that compression takes place “on the fly”
 - Prior versions would create the initial file and then run the compression
 - Which required yet another pass through the data and additional disk I/O



SAS Compress Basics

- Even with Version 9, compression can make your process run slower
- You are trading reduced storage space for increased CPU
- With some forms of compression, you can reduce I/O time
 - Less data is being read
 - I have seen this demonstrated with other tools
- SAS Compression seems to single threaded
 - Same CPU that is performing your process is performing the compression
- SAS Compression may not be the most space efficient
 - UNIX/Linux and Windows compression tools may save more space
 - There will be increased code complexity to use those tools
 - You may save elapsed time since they can run in a separate thread



SAS Compress Basics

- **Compress=Yes**
 - Same as Compress=Char
- **Compress=No**
 - Disables Compression even if options are set
- **Compress=Binary**
 - Heaviest Compression, Highest CPU usage, Highest space savings
- Can also set via Options at system level, command line, in program, or, as will be shown, within the dataset.
- Proc Options result for system I ran these on:
 - **COMPRESS=BINARY** Specifies the type of compression to use for observations in output SAS data sets.



SAS Compress – Simple Write Example

- An example to compare results:

```
libname test "/just/some/directory";
%macro RandBetween(min, max); (&min + floor((1+&max-&min)*rand("uniform")))
%mend;

data test.test_no (compress=no drop=text1-text44) test.test_yes (compress=yes drop=text1-text44)
  test.test_char (compress=char drop=text1-text44) test.test_bin (compress=binary drop=text1-text44);

  array text[44] $20 ( /* 44 different words and phrases */);

  format longstring $200. ;

  DO indexvariable=1 TO 20000000;
    word1=text[%RandBetween(1,44)];
    num1=%RandBetween(1,999999999);
    word2=text[%RandBetween(1,44)];
    num2=rand("uniform");
    word3=text[%RandBetween(1,44)];
    word4=text[%RandBetween(1,44)];
    num3=%RandBetween(1,999999999);
    word5=text[%RandBetween(1,44)];
    num4=rand("uniform");
    num5=%RandBetween(1,999999999);
    word6=text[%RandBetween(1,44)];
    num6=rand("uniform");

    stringlength=%RandBetween(1,179); /* build a random length string */
    longstring=trim(text[%RandBetween(1,44)]);
    do while (length(longstring) < stringlength);
      longstring=trim(longstring)||" "||text[%RandBetween(1,44)];
    end;

    num7=%RandBetween(1,999999999);
    word7=text[%RandBetween(1,44)];

    output test.test_no; output test.test_yes; output test.test_char; output test.test_bin;
  END;
run;
```



SAS Compress – Simple Write Example

- Individual File Size Results:

11:38:58 test_bin.sas7bdat.lck	4907139072
11:38:58 test_char.sas7bdat.lck	5317066752
11:38:58 test_no.sas7bdat.lck	8326414336
11:38:58 test_yes.sas7bdat.lck	5317066752
11:38:59 test_bin.sas7bdat.lck	4914216960
11:38:59 test_char.sas7bdat.lck	5324668928
11:38:59 test_no.sas7bdat.lck	8338407424
11:38:59 test_yes.sas7bdat.lck	5324734464
11:39:00 test_bin.sas7bdat	4920377344
11:39:00 test_char.sas7bdat	5331353600
11:39:00 test_no.sas7bdat	8348631040
11:39:00 test_yes.sas7bdat	5331353600
11:39:01 test_bin.sas7bdat	4,920,377,344
11:39:01 test_char.sas7bdat	5,331,353,600
11:39:01 test_no.sas7bdat	8,348,631,040
11:39:01 test_yes.sas7bdat	5,331,353,600

- We can see that the files grow together – compression is no longer a separate step



SAS Compress – Simple Write Example

- Individual File Results:

```
NOTE: The data set TEST.TEST_NO has 20000000 observations and 17 variables.
NOTE: The data set TEST.TEST_YES has 20000000 observations and 17 variables.
NOTE: Compressing data set TEST.TEST_YES decreased size by 36.14 percent.
       Compressed is 81349 pages; un-Compressed would require 127389 pages.
NOTE: The data set TEST.TEST_CHAR has 20000000 observations and 17 variables.
NOTE: Compressing data set TEST.TEST_CHAR decreased size by 36.14 percent.
       Compressed is 81349 pages; un-Compressed would require 127389 pages.
NOTE: The data set TEST.TEST_BIN has 20000000 observations and 17 variables.
NOTE: Compressing data set TEST.TEST_BIN decreased size by 41.06 percent.
       Compressed is 75078 pages; un-Compressed would require 127389 pages.

NOTE: DATA statement used (Total process time):
      real time           8:22.39
      user cpu time       2:52.89
      system cpu time     26.94 seconds
      memory             1516.40k
      OS Memory          21152.00k
      Timestamp           04/17/2017 12:05:00 PM
      Step Count          265   Switch Count   222
      Page Faults         0
      Page Reclaims       426
      Page Swaps          0
      Voluntary Context Switches 623546
      Involuntary Context Switches 128208
      Block Input Operations 0
      Block Output Operations 0
```



SAS Compress – A Warning

- With small files, compress can make the file larger
- In this case, running the example code for only 20 observations:

Size	File
131,072	test_no.sas7bdat
196,608	test_yes.sas7bdat
196,608	test_char.sas7bdat
196,608	test_bin.sas7bdat

- Even without actual compression, the file size is larger
- SAS Warns you in the log with a NOTE:

```
NOTE: The data set TEST.TEST_NO has 20 observations and 17 variables.  
NOTE: The data set TEST.TEST_YES has 20 observations and 17 variables.  
NOTE: Compressing data set TEST.TEST_YES increased size by 100.00 percent.  
       Compressed is 2 pages; un-compressed would require 1 pages.  
NOTE: The data set TEST.TEST_CHAR has 20 observations and 17 variables.  
NOTE: Compressing data set TEST.TEST_CHAR increased size by 100.00 percent.  
       Compressed is 2 pages; un-compressed would require 1 pages.  
NOTE: The data set TEST.TEST_BIN has 20 observations and 17 variables.  
NOTE: Compressing data set TEST.TEST_BIN increased size by 100.00 percent.  
       Compressed is 2 pages; un-compressed would require 1 pages.
```

-



SAS Compress – Read Example

- Read Times will Vary based on compression method
- In each case, the read code is the same except for the input table
- Uncompressed Read (baseline):

```
libname test "/just/some/directory";  
data null;  
set test.test_no; /* Different datasets for each test */  
    retain total 0;  
    total=total+num1;  
run;
```

NOTE: There were 20000000 observations read from the data set TEST.TEST_NO.

NOTE: DATA statement used (Total process time):

real time	4.99 seconds
user cpu time	1.22 seconds
system cpu time	3.43 seconds
memory	920.25k
OS Memory	21152.00k



SAS Compress – Read Example

- **Compress=Char and Compress=Yes produced similar results:**

NOTE: There were 20000000 observations read from the data set TEST.TEST YES.

NOTE: DATA statement used (Total process time):
real time 12.56 seconds
user cpu time 9.93 seconds
system cpu time 2.52 seconds
memory 1137.56k
OS Memory 21152.00k

- **Compress=Binary used more resources:**

NOTE: There were 20000000 observations read from the data set TEST.TEST BIN.

NOTE: DATA statement used (Total process time):
real time 24.18 seconds
user cpu time 21.75 seconds
system cpu time 2.25 seconds
memory 1151.34k
OS Memory 21152.00k



SAS Compress – Read Example

- A quick comparison:

Example	Elapsed	System	User	Memory
None	4.99 sec	3.43 sec	1.22 sec	920.25k
Yes	12.56 sec	2.52 sec	9.93 sec	1137.56k
Char	12.68 sec	2.44 sec	10.11 sec	1137.25k
Binary	24.18 sec	2.25 sec	21.75 sec	1151.34k



gzip Compression

- GNU Zip (gzip and gunzip) commands
 - Are available on most systems including UNIX, Windows, and Linux (by default).
 - WinZip is available under Windows (and can be read by gzip)
 - Some UNIX zip can read WinZip files

- Significant improvement in space usage:

	size before	size after: gzip fastest	size after: gzip default	size after: gzip max
test_bin	4,920,377,344	3,053,723,102	2,794,141,358	2,780,018,371
test_char	5,331,353,600	2,036,590,374	1,814,911,246	1,796,243,109
test_no	8,348,631,040	2,120,174,601	1,758,621,239	1,737,218,569
test_yes	5,331,353,600	2,036,590,374	1,814,911,246	1,796,264,146

- You actually get less compression on files SAS has already compressed
- The SAS compression interfered with the gzip algorithm



gzip Compression

- There Ain't No Such Thing As A Free Lunch (TANSTAAFL: Robert A. Heinlein)
- The space savings comes at a cost:

	Zip fastest CPU	Unzip fastest CPU	Zip default CPU	Unzip default CPU	Zip max CPU	Unzip Max CPU	Average
test_bin	143.7	59.2	358.7	52.1	803.7	51.8	244.8
test_char	92.0	46.6	281.2	43.0	627.7	43.0	188.9
test_no	108.3	63.5	293.2	55.0	755.4	54.2	221.6
test_yes	92.4	46.4	281.5	43.4	592.5	43.0	183.2
Average	109.1	53.9	303.6	48.4	694.8	48.0	

- And a significant cost in elapsed time:

	Zip fastest ET	Unzip fastest ET	Zip default ET	Unzip default ET	Zip max ET	Unzip Max ET
test_bin	04:04.2	02:48.0	07:56.2	02:11.5	15:26.2	02:14.2
test_char	02:41.8	02:44.7	06:03.5	02:04.8	12:00.2	02:09.1
test_no	03:04.3	03:44.2	06:13.3	02:56.9	14:06.9	03:07.9
test_yes	02:44.4	02:40.7	06:10.7	02:08.7	11:25.5	02:11.1

- But there are ways to reduce these costs



Compression Comparison

- Compression in any form makes sense when:
 - Space is at a premium (just about always)
 - File sizes are large
 - Processing cost is high (data isn't just being read and reported)
- SAS Compression makes more sense when:
 - Processing time is important
 - Want simplicity of code
 - Want immediate access to data
- gzip makes sense when:
 - File is infrequently used – especially when it is kept because you're afraid to get rid of it (or regulatory requirements)
 - Maximum space savings is important
 - File sizes are really large



Taking Advantage of Parallelism – Piping

- You can take advantage of multiple CPU/cores to process compressed data through the use of Pipes.
- SAS supports piping natively for flat files
- SAS requires operating system support for "named pipes"
- Makes use of the "Sequential Data Engine" – often referred to as the "TAPE" engine.
 - You can only write one dataset to it
 - You can only read once
 - proc contents information limited (no 'NOBS' for instance)
 - You can't do both at the same time



Taking Advantage of Parallelism – Piping

- Let's start with an example – minor changes to the earlier Compression Write:

```
libname test "/just/some/directory/base_no_fifo";
/* In UNIX Command Line, execute: mknod /just/some/directory/base_no_fifo p */

%macro RandBetween(min, max);    (&min + floor((1+&max-&min)*rand("uniform")))
%mend;

X "gzip < /just/some/directory/base_no_fifo > /just/some/directory/base6_no_via_fifo.sas7bdat.gz
&";

data test.test_no (compress=no drop=text1-text44) ;
  array text[44] $20 (/* list of 44 words or phrases */);

  format longstring $200. ;

  DO indexvariable=1 TO 20000000;
    /* Nothing changed here */
    output test.test_no; /* Only creating one this time */
  END;

run;

/* These will not work; I'll explain why!
proc print data=test.test_no (obs=10);
run;

proc contents data=test.test_no; run;
*/
```



Taking Advantage of Parallelism – Piping

- Minor changes to the earlier Compression Read example:

```
libname test "/just/some/directory/base_no_fifo";  
/* In UNIX Command Line, execute: mknod /just/some/directory/base_no_fifo p */  
  
X "gunzip --stdout /just/some/directory/base6_no_via_fifo.sas7bdat.gz >  
/just/some/directory/base_no_fifo &";  
  
data _null_;  
set test.test_no;  
    retain total 0;  
    total=total+num1;  
run;
```



Taking Advantage of Parallelism – Piping

- Timing Results:

	zip CPU	unzip CPU	Zip ET	Unzip ET	pipe zip CPU	pipe unzip CPU	pipe zip ET	pipe unzip ET	File Size
gzip Max	755.40	54.20	14:06.9	03:07.9	61.22	5.00	10:33.0	48.14	1,737,218,569
gzip Default	293.20	55.00	06:13.0	02:57.0	59.20	5.11	04:50.9	01:01.9	1,758,621,239
gzip Min	108.30	63.50	03:04.3	03:44.0	59.35	5.12	03:49.0	58.03	2,120,174,601
cat					64.23	5.09	01:34.0	11.03	8,348,631,040
Direct Read					61.08	4.65	02:44.5	4.99	8,348,631,040

- I've included a Direct Read for comparison purposes
- Note that SAS does not report the gzip/gunzip CPU usage
 - Separate Process
 - Separate CPU/Core/Thread
- There are times you can get a "nearly free" lunch.



Taking Advantage of Parallelism – Piping

- What are Pipes?
 - Very similar to the water pipes in your home
 - There is a pump and faucet
 - You are able to pick the direction
 - Data can only flow one way at a time
 - Data can only flow when the pipe program is executing
 - There is a creator and consumer
 - In the Write Example, SAS is the pump, gzip is the faucet
 - In the Read Example, gzip is the pump, SAS is the faucet
 - Data is not stored in the pipe itself
 - May be a bit buffered on disk or may entirely be in memory
 - Won't typically cross networks



Taking Advantage of Parallelism – Piping

- What are Pipes?
 - Requires an entry on disk
 - Created via the mknod (make node) or mkfifo (make first-in first-out):

```
mknod /just/some/directory/base_no_fifo p
```

```
mkfifo /just/some/directory/base_no_fifo
```
 - Pipes (the infrastructure) remain around unless removed
 - Disk entry will look like (using `ls -al` command)

```
prw-rw-r-- 1 MYID my_group_name      0 Apr 02 09:48 base_no_fifo
```

 - "p" tells you this is a Pipe
 - "0" tells you it isn't holding any data
 - You can also run the external command in a script or by hand
 - Useful if X Command not allowed
 - Will not work in Grid environment



Taking Advantage of Parallelism – Piping

- Why won't they work?

- In the Pipe Compression Write I included:

```
/* These will not work; I'll explain why!  
proc print data=test.test_no (obs=10); run;  
proc contents data=test.test_no; run;  
*/
```

- In the program, Libname *test* is a pipe.

- Data flowed through that pipe, and having flowed, is no longer available.

- At least not in this context
 - The data is still available on the disk (written out by gzip)
 - But not to this program unless we reprime, and in this case, reverse the pump:

```
X "gunzip --stdout /just/some/directory/base6_no_via_fifo.sas7bdat.gz > /just/some/directory/base_no_fifo &";  
proc print data=test.test_no (obs=10); run;  
  
X "gunzip --stdout /just/some/directory/base6_no_via_fifo.sas7bdat.gz > /just/some/directory/base_no_fifo &";  
proc contents data=test.test_no; run;
```



Taking Advantage of Parallelism – Piping

- Common Error:

- Attempting to write multiple datasets to (or read multiple from) a sequential library

```
output test.test_no test.test_yes test.test_char test.test_bin;
```

- Will result in an error:

```
ERROR: Attempt to open two sequential members in the same sequential library. File TEST.TEST_YES.DATA cannot be opened.
```

```
NOTE: The SAS System stopped processing this step because of errors.
```

```
WARNING: The data set TEST.TEST_NO may be incomplete. When this step was stopped there were 0 observations and 17 variables.
```



Taking Advantage of Parallelism – Piping

- External Command Example – Write:

- UNIX/Linux commands:

```
mknod mypipe p
gzip mypipe > input.gz & /* runs in
background/parallel */
sas writepipe.sas
```

- writepipe.sas Program:

```
libname test "mypipe";

%macro RandBetween(min, max);      (&min + floor((1+&max-&min)*rand("uniform")))
%mend;

/* X command removed */
data test.test_no (compress=no drop=text1-text44) ;
  array text[44] $20 (/* list of 44 words or phrases */);

  format longstring $200. ;

  DO indexvariable=1 TO 20000000;
    /* Nothing changed here */
    output test.test_no;

  END;
run;
```



Taking Advantage of Parallelism – Piping

- External Command Example – Read:

- UNIX/Linux commands:

```
mknod mypipe p /* not needed if created before)
gzip --stdout input.gz > mypipe & /* runs in background/parallel
*/
sas readpipe.sas
```

- readpipe.sas Program:

```
libname test "mypipe";
/* X command removed */

data _null_;
set test.test_no;
retain total 0;
total=total+num1;
run;
```



Taking Advantage of Parallelism – Piping

- No real timing differences between external and internal (X) command approaches
- Minor Advantages for External Commands:
 - Can trap errors within the gzip command
 - Missing file for instance
 - Control at the shell level
 - Same SAS program able to work for different files
- Minor Disadvantages for External Commands:
 - Increased code complexity
 - Both SAS and UNIX/Linux code required
- Major Disadvantage for External Commands:
 - External command difficult to implement in Grid environment



Caution

- In any commands in this presentation, the single and double quotation marks should be simple, not the “smart quotes” forced by Microsoft. The same applies to dashes or minus signs – they should not be “em dashes” (- versus –)



Wrap Up

?! ! ?!

?

Questions

?

and

?

Answers

?! ! ?!



Filename Piping

- If we have some extra time...
- It is possible to process INFILE or FILE with pipes
 - Much like process with set or data
 - Can be used with Internal or External commands
- SAS also supports the PIPE keyword on the FILENAME statement to allow piping in/out data:
 - FILENAME fileref PIPE 'UNIX-command' <options>;
- Your INFILE or FILE command will include the fileref. Whatever you INPUT or PUT in that data step will involve the specified UNIX command.



Filename Piping

- A Writing Example (should look fairly familiar by now):

```
filename testref PIPE "cat > /just/some/directory/output.txt";

%macro RandBetween(min, max); (&min + floor((1+&max-&min)*rand("uniform")))
%mend;

data _null_;
    file testref;

    array text[44] $20 (/ * 44 words and phrases */);
    format longstring $200. ;

    DO indexvariable=1 TO 200;
        word1=text[%RandBetween(1,44)];
        num1=%RandBetween(1,9999999999);
        word2=text[%RandBetween(1,44)];
        num2=rand("uniform");
        word3=text[%RandBetween(1,44)];
        word4=text[%RandBetween(1,44)];
        num3=%RandBetween(1,9999999999);
        word5=text[%RandBetween(1,44)];
        num4=rand("uniform");
        num5=%RandBetween(1,9999999999);
        word6=text[%RandBetween(1,44)];
        num6=rand("uniform");

        stringlength=%RandBetween(1,179);
        longstring=trim(text[%RandBetween(1,44)]);
        do while (length(longstring) < stringlength);
            longstring=trim(longstring)||" " || text[%RandBetween(1,44)];
        end;

        num7=%RandBetween(1,9999999999);
        word7=text[%RandBetween(1,44)];

        put word1 num1 word2 num2 longstring;

    END;
run;
```



Filename Piping

- A Reading Example (should look fairly familiar by now):

```
filename testref PIPE "cat /just/some/directory/output.txt";  
  
data out;  
    infile testref;  
    input name $;  
run;  
  
proc print data=work.out (obs=10); run;
```

- Produces the following

Obs	name
1	with commas 63344454
2	and enclose 58066050
3	or double 882972945
4	of an array 97957098
5	To do 368188872 init
6	and enclose 19271463
7	and enclose 90992099
8	or spaces 8165156291
9	with commas 42546153
10	or spaces 96397033 i



Filename Piping

- I can also use FILENAME ZIP. If I need a file in GZIP format, I can use the GZIP modifier – but only if I have version 9.4M5 or newer. If I have an earlier version, I have to use a FILENAME PIPE:

```
filename gzipit zip '/my/output/directory/file.txt.z'; * run the UNIX zip;
```

```
filename gzipit zip '/my/output/directory/file.txt.gz' gzip; * run the UNIX gzip;
```



References 1

- **NOTES**

- Indexing and Compressing SAS® Data Sets:
<http://www2.sas.com/proceedings/sugi28/003-28.pdf>
- SAS(R) 9.2 Language Reference: Dictionary, Fourth Edition:
<http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a001288760.htm>
- Programming Tricks For Reducing Storage And Work Space:
<http://www2.sas.com/proceedings/sugi27/p023-27.pdf>
- How to Reduce the Disk Space Required by a SAS® Data Set:
<http://www.lexjansen.com/nesug/nesug06/io/io18.pdf>



References 2

- **NOTES**

- Accessing Sequential-Format Data Libraries (pipes):
<http://technology.msb.edu/old/training/statistics/sas/books/unix/z0386494.htm>
- Smokin' With UNIX Pipes (FILENAME):
<http://www2.sas.com/proceedings/sugi25/25/cc/25p103.pdf>
- SAS® 9.4 Companion for UNIX Environments, Sixth Edition (X command):
<http://support.sas.com/documentation/cdl/en/hostunix/69602/PDF/default/hostunix.pdf>
- Using SAS with Pipes or as a Filter under UNIX:
<https://www.linkedin.com/pulse/using-sas-pipes-filter-under-unix-david-horvath?published=t>



To COMPRESS or Not

The Author can be contacted at:

David B. Horvath, CCP
504 Longbotham Drive, Aston PA 19014-2502, USA
Phone: 1-610-859-8826
Email: dhorvath@cobs.com
Web: <http://www.cobs.com/>
LI: <http://www.linkedin.com/in/dbhorvath>

All trademarks and servicemarks are the property of their respective owners.



Wrap Up (For Real)

?! ! ?!

?

Questions

?

and

?

Answers

?! ! ?!



Compression References

- My Word/Phrase array:

```
array text[44] $20 ('For some' 'applications' 'it can be'  
    'beneficial' 'to assign' 'initial' 'values to the' 'variables or'  
    'elements' 'of an array' 'at the' 'time that' 'the array' 'is  
defined'  
  
    'To do' 'this' 'enclose' 'the initial' 'values in' 'parentheses'  
    'at the end' 'of the' 'ARRAY' 'statement' 'Separate' 'the values'  
    'either' 'with commas' 'or spaces' 'and enclose' 'character'  
    'values in' 'either single' 'or double' 'quotation' 'marks'  
    'The following' 'statements' 'illustrate' 'the' 'initialization'  
    'of numeric' 'and' 'character values');
```

