Paper 3329-2015

Efficiently Using SAS® Data Views

Shigui Weng, Ya-Jiun Tsai, and Shy DeGrace, United States Census Bureau

ABSTRACT

For the Research Data Centers of the United States Census Bureau, the demand for disk space substantially increases with each passing year. Meanwhile, providing data access to our researchers in a reliable, efficient, and timely manner is a great challenge for us. In addition, more demands from researchers and data providers will come in the near future as the Research Data Centers have just expanded to the Federal Statistical Research Data Centers to provide data access to researchers not just for the Census Bureau but also for multiple government agencies. SAS data views provide an alternative way to deploy data sets in our data repository. Efficiently using the SAS® data view might successfully address these challenges within the Research Data Centers. This paper discusses the usage and benefits of SAS data views, and study the disk space and time efficiency of using SAS Data Step views with regular ASCII files, compressed ASCII files, and compressed SAS data sets.

INTRODUCTION

The Census Research Data Centers (RDCs) provide access to restricted data for approved projects for statistical purposes at the 18 locations in the United States. All data files in the RDC network are hosted and managed centrally at the Census Bureau's Bowie Computing Center. For the RDCs, the demand for disk space substantially increases with each passing year. Meanwhile, providing data access to our researchers in a reliable, efficient, and timely manner is a great challenge for us. In addition, more demands from researchers and data providers will come in the near future as RDCs have expanded to become the Federal Statistical Research Data Centers. The RDCs now provide data access to researchers for the Census bureau as well as other government agencies.

SAS datasets are the standard data format provided to researchers in the RDCs. SAS data views provide an alternative way to deploy our datasets in our data repository. We need to find an optimal method of providing data access that meets requirements of researchers and data providers, while using available disk space efficiently.

SAS data views have been available in the SAS system for years. More and more SAS users realize the importance and advantages of SAS data views. A SAS view is a virtual SAS data set that reads data values from other data files. It contains no data, but only variable definitions and other related information to read data values. Files from which SAS views read data are called underlying data files. SAS data views use very little disk space, but can still present data from underlying data files to SAS users. Reading and processing data directly from underlying data files can save significant disk space. For SAS users, SAS data views are transparent and can be used as regular SAS datasets when created properly.

SAS data views can hide complex data processing and analysis form users. They can also present selected statistical reports or subsets of the underlying data files. SAS data views always provide up-to-date data access to underlying data files, as data are processed and derived upon execution. SAS data views can minimize copies of the underlying data, which saves the time and effort of maintaining multiple copies of datasets.

When compared with traditional SAS datasets, however, a major disadvantage of SAS data views is the overhead to read and process underlying data. The decision to employ SAS data views should be determined by disk space usage, data use frequency, overhead time, and other related factors.

Any opinions and conclusions expressed herein are those of the authors and do not necessarily represent the views of the U.S. Census Bureau. All results have been reviewed to ensure that no confidential information is disclosed.

Our study focuses specifically on our current computing environment, which consists of Linux clusters with multiple Intel Six-Core X5670 CPUs running the Red Hat Enterprise Linux Server and SAS® 9.2 or 9.4.

A SAS data view can be a SAS DATA step view (created with a SAS DATA step), a SAS SQL view (created with PROC SQL), or a SAS/ACCESS view (created with SAS/ACCESS software). This paper will not discuss the SAS/ACCESS view. The terms “SAS data view” and “SAS view” are interchangeable in this paper.

We will first discuss how to use SAS data views. Then we will discuss and analyze disk usage and the overhead time involved using SAS DATA step views.

CREATING AND USING SAS DATA VIEWS

We created and used SAS DATA step views and SAS PROC SQL views with the following underlying data files:

- SAS data sets
- ASCII files
- Gzip-compressed ASCII files

SAS DATA STEP VIEWS WITH EXTERNAL SAS DATA SETS

We often use SAS DATA step views to minimize multiple copies of existing SAS data sets. To create a DATA step view, we use the ALTER option and the SOURCE=ENCRYT option to protect the view from unauthorized replacement and to hide the source code from SAS users.

The example below creates a SAS DATA step view that merges data from two existing SAS data sets, which are Public Use Microdata Sample (PUMS) file from the American Community Survey (ACS). The source code is protected with the ALTER and SOURCE options:

```
/*Program 1*/
libname acsp "acs/2013/unix_pus";
libname acsh "acs/2013/unix_hus";
libname acs  "acs/2013";

data acs.psam_phus /view=acs.psam_phus (alter=”Census” source=encrypt);
merge acsp.psam_pus (in=p) acsh.psam_hus;
by serialno;
if p;
run;
```

The above code creates a SAS data view file in the acs library merging the housing file with the person file. Note that the two underlying SAS data sets must be sorted before using the view.

To see the SAS view source code, use the DESCRIBE statement with the ALTER option,

```
/*Program 2*/
libname acs  "acs/2013";
data view=acs.psam_phus (alter="Census");
describe;
run;
```

Without providing the proper ALTER password, users cannot see the protected source code of the view and cannot replace the view in the SAS session. They can, however, still see the source code with the proper operating system (OS) commands if the SOURCE=ENCRYPT option was not used when the SAS data view was created.

Note that users are still able to delete the SAS view file with the OS command if they have adequate privileges. System or data administrators should set the proper permission and access controls for the SAS view file at the OS level.

PROC SQL VIEWS WITH EXTERNAL SAS DATA SETS

We use PROC SQL views to reduce multiple copies of existing SAS data sets when SQL views have advantages over the SAS DATA step views. An example of this would be merging of datasets that are not sorted or indexed by key variables. The example code below applies if the housing dataset and the person dataset are not sorted by the serialno variable in the above example (Program 1):

```
/*Program 3*/
libname acsp "acs/2013/unix_pus";
libname acsh "acs/2013/unix_hus";
libname acs  "acs/2013";
proc sql;
create view acs.psam_phus_svew as
 select *
  from acsp.psam_pus p left join acsh.psam_hus h
    on p.serialno=h.serialno
    order by serialno;
run;
```

Index files are very useful for quick observation access, but require too much disk space and significant maintenance time and effort. In our data repository, we have stopped maintaining index files for some data products, especially for rarely used SAS data sets. Instead, we use PROC SORT to create sorted SAS data sets or create PROC SQL views when necessary.

We also use PROC SQL views to present some commonly used summary statistics. The example below shows the age distribution from the ACS PUMS data:

```
/*Program 4*/
libname acsp "acs/2013/unix_pus";

proc sql;
create view acsp.agesex as
 select agep label='Age Group', sex, count(*) as count label="Count"
  from acsp.psam_pus
  group by agep, sex;
  ……
  complex process statements;
  ……
run;
```

Using the SQL view to present the summary statistics always shows the up-to-date results of the underlying data. In addition, the SQL view can also hide a complex code from users.

ACCESSING AND SUBSETTING ENCRYPTED SAS DATA SETS

SAS passwords can be used to control access to SAS data sets within SAS sessions. However, the password protection is limited. Unauthorized users with adequate OS privileges can still read the password protected SAS data sets using external programs.

SAS data file encryption provides a solution to protect SAS data sets. Combining the data file encryption technique with the proper OS level permission can provide better and stronger protection for restricted access files.

Using SAS DATA step views can simplify access to encrypted SAS datasets while hiding passwords and the encryption key from users. In addition, a SAS DATA step view can present subsets of underlying data

sets by dropping or keeping observations and/or sensitive variables. The subsets can be designed to correspond to the specific access control level of users. The steps include:

- Encrypt underlying data sets
- Create DATA step views
- Set up the proper permission at the OS level to protect the underlying data sets and the DATA step views

Encrypting Underlying Data Sets

SAS data sets can be encrypted as indicated in the example below:

```
/*Program 5*/
libname libx  "library8/plib5";
libname libin "library8/unencrypt";

data libx.surveyx (encrypt=yes alter=DAdmin read=DUser);
set libin.surveyx;
run;
```

The above passwords are simple. However, more complex passwords or SAS-encoded passwords (PROC PWENDE) can be used to provide stronger protection. Once the encryption is complete, the unencrypted dataset should be removed, and the passwords should be kept in a safe place.

Creating SAS DATA step views

Following the encryption of the underlying SAS data sets, SAS DATA step views are defined to read the encrypted data file. A DATA step view can be tailored to only read and process a subset of encrypted underlying data files. The defined DATA step views should be protected with at least the SOURCE=ENCRYPT option and the ALTER option. The example below contains four SAS DATA step views:

```
/*Program 6*/

libname libx  "library8/plib5";

/*Project 302 is authorized to use the whole dataset*/
data libx.surveyx302 /view=libx.surveyx302(alter=DAdmin read=pw302
source=encrypt);
set libx.surveyx(read=DUser);
run;

/*Project 110 is authorized to use everything but date of birth (bdate)*/
data libx.surveyx110 /view=libx.surveyx110(alter=DAdmin read=Ec981
source=encrypt);
set libx.surveyx(read=DUser);
drop bdate;
run;

/*Project 553 is authorized to use some states (stcode >=20),*/
/*and all variables except bdate, pik1 and pik2*/
data libx.surveyx553 /view=libx.surveyx553(alter=DAdmin read=Va550
source=encrypt);
set libx.surveyx(read=DUser);
where stcode >= 20;
drop bdata pik1 pik2;
run;
```

```
/*Surveyx group access,no read password*/
data libx.surveyxgrp /view=libx.surveyxgrp(alter=DAdmin source=encrypt);
set libx.surveyx(read=DUser);
where stcode >= 20;
drop bdata pik1 pik2 income stcode education age;
run;
```

Each view defines the access scope of a project or group. The first view allows the members of Project 302 to use all variables and observations of the underlying dataset. The second view allows the members of Project 110 to access everything except date of birth (bdate). The third view allows the members of Project 553 to access the observations where the state code is 20 or larger and all variables except bdate, pik1, and pik2. The last view is for the surveyx group access (controlled by the OS permission), allowing the group to access the observations where the state code is 20 or larger and dropping the bdate, pik1, pik2, income, stcode, education, and age variables.

Setting Up Proper OS Permission

When the underlying dataset and the SAS DATA step views are protected with passwords and encryption, users are unable to read the data or see the source code without proper passwords. They can still delete or replace the files using the OS commands if the OS level protection is not established.

The ability to execute the above SAS views in Program 6 is determined by both the OS permission and the SAS passwords. For the first three views, users require both the OS permission and the SAS READ password. For the forth view, users need only the OS permission with the view file since no read password is defined.

The OS permission for the underlying data file and the four views created in Program 6 can be established with the following settings:

- Projects 302, 110, 553, and the surveryx group should have read-only access to the underlying dataset (surveyx.sas7bdat).
- Each project or group should only have read access to the view. For example, Project 553 should have access (read-only) to surveyx553.sas7bvew, and the surveyx group should only have read-only access to the surveyxgrp.7sasbvew.
- Only data administrators should have read and write access to the underlying data and all four view files.

Executing SAS DATA step views

In the fourth example above, the view is protected at only the OS level, and the surveyx group members do not need the READ password to execute the data view. However, the members of Projects 302, 110, and 553, can execute the data view with the proper password as the example below:

```
/*Program 7*/
libname libx  "library8/plib5";
proc print data=libx.surveyx553(read=Va550);
var serialno age--pik3;
run;
```

USING SAS DATA VIEWS WITH EXTERNAL ASCII FILES

The SAS system has the capability to read various formats of external ASCII data files. In addition, using the filename statement with PIPE extends its ability to process gzip-compressed ASCII files prior to the DATA step processing.

We created SAS data views that directly read ASCII or gzip-compressed ASCII files without creating permanent SAS datasets to save disk space.

The example below creates a SAS DATA step view from the external ASCII file (a subset of ACS PUMS):

```
/*Program 8*/

/*the uncompressed ASCII file*/
filename acsdat "acs/2013/unix_pus/basic_data.dat";

/*the Gzip compressed file*/
*filename acsdat pipe "gunzip –c acs/2013/unix_pus/basic_data.dat.gz";

libname  acslib "library8/acslib";

data acslib.acs1 /view=acslib.acs1 (alter=DAdmin source=encrypt);
infile acsdat lrecl=20;
input serialno $ 1-9 sporder  10-11  puma $ 12-16
      agep  17-18    mar $ 19        sex $ 20 ;
length sporder 3 agep 3 ;
run;
```

For compressed ASCII files, the filename statement with PIPE was used to decompress the ASCII files upon execution:

```
filename acsdat pipe "gunzip –c acs/2013/unix_pus/basic_data.dat.gz";
```

NOTES FOR LIBNAME AND FILENAME GLOBAL STATEMENTS

In previous examples, the LIBNAME and FILENAME global statements are used. However, SAS DATA step views have the restriction that these global statements cannot be saved to the DATA step view. This holds true even if these global statements are included in the source code of the view. In other words, the libref or fileref defined by LIBNAME or FILENAME is not part of the DATA step view. The LIBNAME and/or FILENAME statement must be used prior to the execution of the DATA step view. For example, the SAS system displays errors in the log file if the following code is executed without FILENAME and LIBNAME statements:

```
5        /*Program 9, executes the SAS views created in Program 8*/
8
9        /*no fileref and libref defined*/
10       proc print data=acslib.acs1(obs=100);
ERROR: Libname ACSLIB is not assigned.
11         run;

12
13       /*libref is defined, but fileref not*/
14       libname  acslib "library8/acslib";

15       proc print data=acslib.acs1(obs=100);
ERROR: No logical assign for filename ACSDAT.
ERROR: Failure loading view ACSLIB.ACS1.VIEW.
       Error detected during View Load request.
```

This restriction of the global statements with SAS views could be problematic for users. For regular SAS datasets, users need only the location of the datasets. For SAS views, however, users must know both the locations of SAS views and all underlying data files if they are not in the same locations as the SAS view files. This problem is further compounded with that fact the users must have knowledge of the FILENAME statement with PIPE method. This may prove to be quite difficult for SAS users.

Two possible solutions for this problem exist. The first solution is to define libref and fileref in a system-wide configuration or startup file. The LIBNAME and FILENAME statements may be put into the system-wide autoexec.sas making the required libref and fileref available to all the users before starting SAS sessions. The second solution uses fixed physical locations of underlying data files when creating the SAS DATA step views. The examples below are revised from Program 1 and Program 8:

```
/*Program 10, revised from program 1*/

libname acs  "acs/2013";

data acs.psam_phus /view=acs.psam_phus (alter="Census");
merge 'acs/2013/unix_pus/psam_pus' (in=p) 'acs/2013/unix_hus/psam_hus';
by serialno;
if p;
run;


/*Program 11, revised from program 8*/
libname  acslib "library8/acslib";

data acslib.acs1 /view=acslib.acs1 (alter=DAdmin source=encrypt);
infile 'acs/2013/unix_pus/basic_data.dat' lrecl=20;
input
   SERIALNO $ 1-9 SPORDER  10-11 PUMA $ 12-16
   AGEP  17-18    MAR $ 19       SEX $ 20;
length SPORDER 3   AGEP 3 ;
run ;

/*Program 12*/
libname  acslib "library8/acslib";

data acslib.acs1 /view=acslib.acs1 (alter=DAdmin source=encrypt);
infile "gunzip -c acs/2013/unix_pus/basic_data.dat.gz" PIPE LRECL=20;
input
   SERIALNO $ 1-9  SPORDER  10-11    PUMA $ 12-16
   AGEP  17-18     MAR $ 19          SEX $ 20;
length SPORDER 3  AGEP 3 ;
run ;
```

To execute the above SAS views, users just need to use the LIBNAME statement as they would with a regular SAS dataset. Supplying the LIBNAME and FILENAME statements to reference the underlying data files will not be a concern of the user.

We prefer the second solution because that frees us from maintaining global statements for SAS views.

As in Program 10 and Program 12, the syntaxes of MERGE and INFILE statements are not commonly used. It is even less common to see examples like this in the SAS documentation. We have successfully tested these code examples with SAS® 9.2 and SAS® 9.4 on our Linux cluster.

CREATING INDEX FILES ONLY WHEN NECESSARY

We often see SAS datasets deployed with their index files, even when these files are not necessary. Index files can consume significant amount of disk space. If the PROC SORT procedure or PROC SQL views can merge SAS data sets, there is no need to create index files.

Description	Size MB	Index File Size single variable MB	Index File Size two variables MB
SAS data set (uncompressed)	38033.78	5247.66(13.79%)	13670.78(35.94%)
SAS data set (compress=char)	36474.63	5247.66(14.39%)	13670.78(37.48%)
SAS data set (compress=Binary)	42825.03	5247.66(12.25%)	13670.78(31.92%)

Table 1. SAS Index File Disk Space Usage

Table 1 shows the disk space usage of a SAS dataset in our data repository. The index files consume much more disk space than we originally anticipated. The composite (two or more variables) index file uses at least 36% of the size of the SAS uncompressed data set.

Table 1 also shows that the SAS char and binary compressions do not compress the index files. Additionally, the char compression reduced the file size by only a small percentage, while the binary compression actually increased the file size.

DISK SPACE USAGE ANALYSIS

SAS data views can save disk space and prevent the creation of unnecessary copies of data sets, as well as the need for index files. In this section, the amount of disk space that SAS data views can save will be discussed. In addition, the Disk Space Usage Efficiency (DSUE) will be analyzed. As the SAS DATA step view is our major tool to save disk space, they will be used to conduct our analysis on a frequently used population census file and 48 Quarterly Workforce Indicator public use files (QWIPUF).

We will compare and analyze five types of each data file, including:

- SAS DATA step view with uncompressed ASCII files (VEWASC)
- SAS DATA step view with gzip-compressed ASCII files (VEWGZASC)
- Normal SAS data sets (SASNDS, uncompressed)
- Char-compressed SAS data set (SASCCDS)
- Binary-compressed SAS data sets (SASBCDS)

For convenience, the abbreviations provided in the parentheses above will be used to refer to these file types for the remainder of this paper. VEWASC is the SAS DATA step view that reads the external ASCII file (Program 11), and VEWGZASC is the SAS DATA step view that reads a gzip-compressed ASCII file (Program 12). In our analysis, SASNDS is the baseline to which other types will compare.

DISK SPACE USAGE EFFICIENCY

Table 1 shows the sizes of five types of the population census files. As the size of SAS DATA step view file is only about 4KB, we will ignore it in our analysis.

From our experience, compared with uncompressed SAS datasets, SAS compression can usually reduce file sizes significantly. For this particular file, however, SAS compressions are inefficient. SASCCDS only reduces the size by 4.0% ($\frac{38033.78-36474.63}{38033.78} \times 100\%$) and SASBCDS actually increases the size by 13% ($\frac{42825.03-38033.78}{38033.78} \times 100\%$).

In contrast, VEWGZASC compresses the data quite efficiently. The VEWGZASC size is approximately 4.5GB, which is 11.8% of the original ASCII file size. Compared to SASNDS, VEWGZASC reduces the size by 88.2% and saves about 33.5GB of disk space.

Using SASNDS as a baseline, the DSUE is defined as the percentage of the saved disk space compared to the size of the normal uncompressed SAS data set as follows,

$$DSUE = \left(1 - \frac{size\ of\ all\ involved\ non\ SASNDS\ files}{size\ of\ all\ involved\ SASNDS\ files}\right) \times 100\%$$

For VEWGZASC,

$$DSUE = \left(1 - \frac{4496.69 + 0.041}{38033.78}\right) \times 100\% = 88.2\%$$

Table 2 shows that VEWGZASC is the most efficient method in saving disk space.

File Type Abbreviation	Description	Size (MB)	Disk Space Usage Efficiency (%)
VEWASC	ASCII+ SAS DATA step view	37975.70 0.041	0.1
VEWGZASC	Gzip compressed ASCII+ SAS DATA step view	4496.69 0.041	88.2
SASNDS	SAS data set (uncompressed)	38033.78	0.0
SASCCDS	SAS data set (compress=char)	36474.63	4.0
SASBCDS	SAS data set (compress=Binary)	42825.03	-12.6

Table 2. Disk Space Usage: the Population Census File

FURTHER ANALYSIS ON DISK SPACE USAGE

Compression methods, variable types, and other factors can affect DSUE. SAS binary compression is more efficient for a large data set with many numeric variables (binary data). In order to collect more information from our data repository, the 48 QWIPUFs that we have deployed with SAS DATA step views were analyzed.

Table 3 shows some summary statistics of the 48 files by file type. The average size of VEWGZASC files is approximately 443MB, which is the smallest of the five types. The average sizes of other file types are, in ascending order, SASBCDS (4007MB), VEWASC (4116MB), SASCCDS (5744MB), and SASNDS (11708MB).

Table 3 shows the results similar to the above average file size comparison. The DSUE of VEWGZASC is approximately 96%, and is the largest percentage among five types.

File Type	Description	File Size			Disk Space Usage Efficiency		
		Average (MB)	Minimum (MB)	Maximum (MB)	Mean (%)	Minimum (%)	Maximum (%)
VEWASC	DATA step view with ASCII	4115.51	105.70	16161.49	64.85	60.54	65.89
VEWGZASC	DATA step view with gzip-compressed ASCII	442.79	15.29	1664.10	96.22	94.29	97.11
SASNDS	SAS data set (uncompressed)	11708.47	267.92	46417.06	0.00	0.00	0.00
SASCCDS	SAS data set (compress=char)	5743.80	143.01	22622.61	50.94	46.62	52.15
SASBCDS	SAS data set (compress=Binary)	4006.53	116.63	18121.07	65.78	56.46	61.77

Table 3. Disk Space Usage: the 48 QWIPUF

Unlike the population census files in Table 2, both SASCCDS and SASBCDS work well for the 48 QWIPUF files. The average DSUE of SASCCDS is approximately 51%, and the average DSUE of SASBCDS is approximately 66%. In these cases, the SAS binary compression is more efficient than the SAS char compression.

We believe the difference in data structure is the root cause of the varying results in the SAS compressions between the population census file (Table 2) and the 48 QWIPUFs (Table 3). In our analysis, the data structure includes variable type (numeric or character), length, and other characteristics.

Figure 1 is a scatter plot that depicts the DSUEs of the 48 files by file size and type. It shows that VEWGZASC is the most efficient way to save disk space with files of any size. Other than smaller files, DSUE does not change much when the file size increases.

The DSUE is a percentage and is somewhat stable for most files (especially for large files). It can be used to compare disk usage efficiencies and to estimate disk space usage for files with similar data structures but significantly different sizes.

Compared to SASNDS, VEWGZASC saves 96% disk space on average. For the total size of 562GB of these QWIPUFs. VEWGZASC uses only approximately 21GB (48*443MB) and saves a total of 540GB (48*11708*0.96MB). The 96% of this DSUE is a tremendous disk space saving.

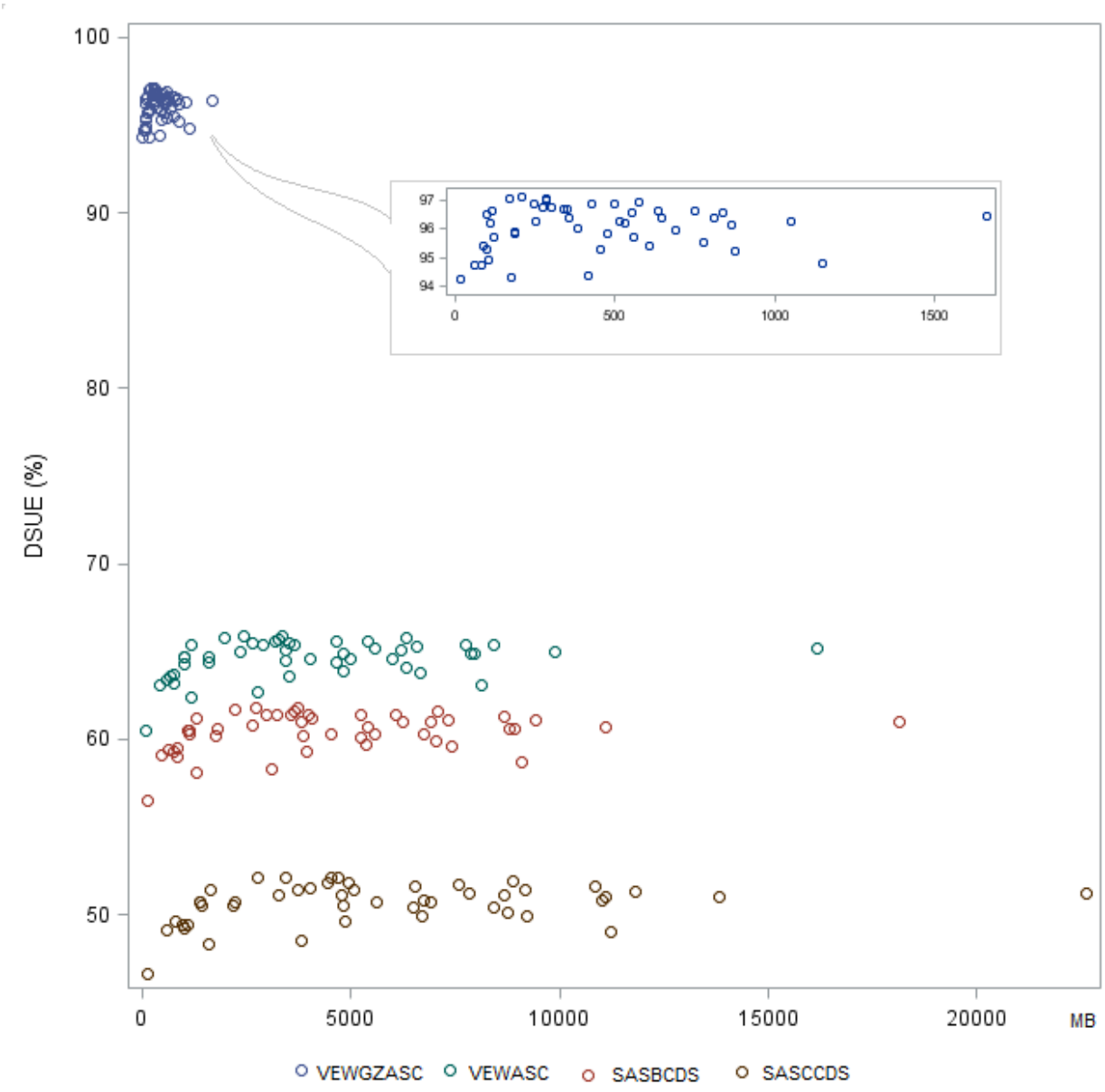

Figure 1. File Size and Disk Space Usage Efficiency by File Type

OVERHEAD TIME ANALYSIS

SASCCDS, SASBCDS, VEWASC and VEWGZASC have an overhead required to process underlying data files when compared to SASNDS. For SASCCDS and SASBCDS, the overhead is to decompress the SAS datasets. For SASASC, the overhead is to read and process the external ASCII files with the DATA step statements defined in the SAS DATA step view. For SASGZASC, the overhead is to decompress the external gzip ASCII files, plus the overhead of SASASC.

The overhead of SAS data views is a primary concern for us. Evaluation of the overhead time is a critical for us to decide whether we should use SAS data views or not.

We used the FULLSTIMER SAS system option to collect the real time, user CPU time, and system CPU time for the PROC FREQ procedure to process underlying data and complete the analysis. The real time is the elapsed time for users to wait for the PROC FREQ procedure to complete. The CPU time is the time that the server system requires to execute the PROC FREQ procedure. In our analysis, the CPU

time is the sum of user CPU time and system CPU time. We collected the data on the least busy node of our cluster servers when the system had fewer users, processes, and I/O activities.

The example below shows the real time, the user CPU time, and the system CPU time from a SAS log file:

```
ibname popcen  'library8/popcen';

proc freq data=popcen.cenus_p;
table qsex;
run;

NOTE: The PROCEDURE FREQ printed page 6.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           16:45.63
      user cpu time       11:57.35
      system cpu time     4:44.15
```

DEFINITIONS OF OVERHEAD TIME AND TIME EFFICIENCY LOSS (TEL)

Similar to the analysis for the disk space usage above, the population census file is used to start our overhead time analysis.

The time that the PROC FREQ procedure requires to process the uncompressed SAS dataset and complete analysis is used as the baseline to calculate the overhead time:

$$Overhead\ Time = Time\ Spent\ on\ nonSASNDS - Time\ spent\ on\ SASNDS$$

The Time Efficiency Loss (TEL) as defined as:

$$TEL = \frac{Overhead\ CPU\ Time}{Total\ CPU\ Time\ Spent\ on\ nonSASNDS} \times 100\%$$

System resources, such as I/O intensive activities, can significantly affect the real time that is not comparable at different time. For this reason, the CPU time is used to calculate TEL.

The TEL is the percentage of the overhead time (extra time) of the total time spent on the SAS DATA step view or SAS compressed data sets in contrast to SASNDS. In Table 4, the total CPU time spent by PROC FREQ on VEWASC is approximately 675.59 seconds, the CPU time of SASNDS is approximately 67.78 seconds, the overhead time is 607.81 seconds, and the TEL for VEWASC is,

$$TEL = \frac{607.81}{675.59} \times 100\% = 89.97\%$$

Data File	Time (Seconds)		Overhead Time (Seconds)		Time Efficiency Loss (%)
	Real	CPU	Real	CPU	
VEWASC	692.76	675.59	595.78	607.81	89.96729
VEWGZASC	1005.63	1001.5	908.65	933.72	93.23215
SASNDS	96.98	67.78	0.00	0.00	0.00
SASCCDS	252.58	240.31	155.6	172.53	71.79477
SASBCDS	166.3	154.63	69.32	86.85	56.16633

Table 4. Real Time, CPU Time, Overhead Time and TEL: the Population Census File

Table 4 shows that VEWGZASC (TEL=93%) uses more time than any of the other types. Its CPU time is approximately 16 minutes (933.72 seconds) longer than SASNDS.

FURTHER ANALYSIS ON OVERHEAD TIME AND TEL

Similar to the disk usage analysis above, an analysis of the 48 data files on CPU time by file size and type was performed to collect more information from files with different data structures. More importantly, this helps us to evaluate our current implementation of SAS DATA step views.

Table 5 shows summary statistics of CPU time, overhead time, and TEL for the 48 data files. The average overhead times in a descending order are VEWGZASC (103.22 seconds), VEWASC (92.50 seconds), SASCCDS (49.44 seconds), and SASBCDS (32.42 seconds). VEWGZASC has the longest overhead time on average, but it is still less than two minutes. In our experience, this is acceptable for our regular statistical analysis and data processing. The maximum overhead time of VEWGZASC is approximate 405 seconds (6 minutes and 45 seconds), which is also acceptable, as this file is 16161 MB in ASCII, 46417 MB in the uncompressed SAS dataset, 22623 MB in the char-compressed SAS dataset, or 18121 MB in the binary-compressed SAS dataset.

For data archiving, we plan to use VEWGZASC to provide SAS users a direct access similar to regular SAS datasets as the overhead time is not a concern for this purpose. Note that maintaining the SAS programs to create VEWGZASC is very important because VEWGZASC is not executable on other platforms.

For infrequently used data files, such as the 48 files we selected for our analysis, we employed VEWGZASC because the overhead time is less of a concern than the disk space usage.

Note that SASBCDS (binary compression) has less average overhead time and less average TEL than SASCCDS or any other type. This indicates that we may need to use the SAS binary compression for deploying SAS datasets in the future.

Table 5 shows that the results of the TELs are similar to those of the overhead times. VEWGZASC has the largest average TEL (92.15%) among the five types. The other average TELs in descending order are VEWASC (91.32%), SASCCDS (84.99%), and SASBCDS (78.81%).

File Type	Time	Mean (sec)	Min (sec)	Max (sec)
VEWASC	CPU time	101.50	2.39	400.47
	Overhead time	92.50	2.17	363.11
	TEL	91.32	90.57	92.76
VEWGZASC	OPU time	112.21	2.68	442.14
	overhead time	103.22	2.46	404.78
	TEL	92.15	91.51	93.54
SASNDS	CPU time	9.00	0.22	37.36
	Overhead time	0.00	0.00	0.00
	TEL	0.00	0.00	0.00
SASCCDS	CPU time	58.43	1.44	229.64
	Overhead time	49.44	1.22	192.28
	TEL	84.99	83.36	87.96
SASBCDS	CPU time	41.42	1.04	162.87
	Overhead time	32.42	0.82	125.51
	TEL	78.81	76.31	83.14

Table 5. Real Time, CPU Time, Overhead Time and TEL: the 48 Files

The overhead time is a direct measurement of time efficiency, but it changes when the file size changes. Similar to the disk space analysis, the relationship between TEL and file sizes should be explored further for the proper use of TEL comparisons.

Figure 2, which shows TELs for four file types (VEWGZASC, VEWASC, SASCCDS, and SASBCDS) in contrast with SASNDS, visually indicates the same result as Table 5. This figure also indicates that the TELs for VEWASC and VEWGZASC do not change much when the file size changes, especially for large data files. TEL could be useful when comparing the time efficiency or estimating overhead time of VEWGZASC between files with the similar data structure. For SASCCDS and SASBCDS, TEL fluctuates when the file size increases and may not be suitable for TEL comparisons between different files sizes.

Compression methods, such as SASBCDS, SASCCDS, and VEWGZASC affect TELs, as shown in Table 5 and Figure 2. In addition, the data structure, such as a variable type (numeric or character), is also an important factor to contribute to TEL. For this reason, TEL comparisons of different files should be performed among similar data structures.

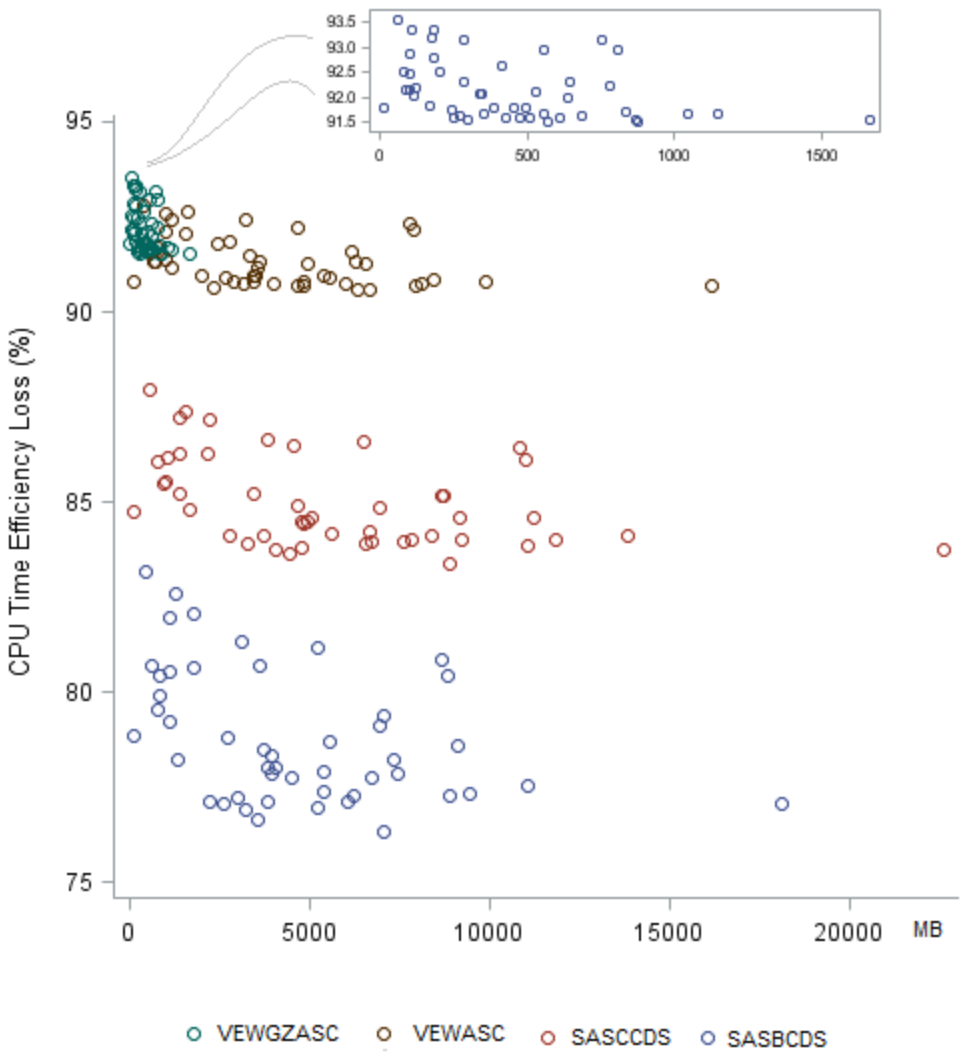

Figure 2. Time Efficiency Loss (TEL) by File Type

Note that the overhead time in our analysis is based on the CPU time. The real time, which is the time for users to wait for the PROC FREQ procedure to complete, could be much longer than the CPU time when the server load is very high.

CONCLUSION

- SAS data views provide an alternative method to deploy data to SAS users.
- SAS data views can save significant disk space. In our analysis of 48 files, VEWGZASC can save about 96% disk space on average compared to SASNDS.
- Proper use of SAS data views can minimize duplication of data files, provide current reports, hide complex process code from users, and reduce the usage of index files.
- Using SAS DATA step views with the SAS file encryption and OS access controls can provide stronger protection for underlying data and different level subset access of the underlying data file.

- Compared with the SASNDS, SAS DATA step views need overhead time to process the underlying data. In our analysis, the average overhead time for VEWGZASC is less than 2 minutes. Even for the largest file, the overhead time of VEWGZASC is about 6 minutes and 45 seconds, which is acceptable for most situations.
- VEWGZASC is recommended for large and infrequently used files as well as for data archiving purposes to provide SAS users with direct and immediate data access.
- DSUE can be used to compare the disk usage efficiency or estimate disk space usage for files that have similar data structures but significantly different sizes.
- TEL can be used to compare the time efficiency or estimate overhead time for VEWGZASC for files that have similar data structures but significantly different sizes.

REFERENCES

SAS ® 9.2 Language Reference: Concepts, Second Edition

ACKNOWLEDGMENTS

Thank Todd Gardner, J. Trent Alexander, and Jacob Enriquez for their excellent input and comments.

CONTACT INFORMATION

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

Shigui Weng
US Census Bureau
shigui.weng@census.gov

Ya-Jiun Tsai
US Census Bureau
ya.jiun.tsai@census.gov

Shy DeGrace
US Census Bureau
shy.degrace@census.gov

For more information regarding RDC and Center for Economic Studies, please visit http://www.census.gov/ces.