SAS® GLOBAL FORUM 2020 #SASGF

SAS4147-2020

Best Practices for Converting SAS® Code to Leverage SAS® Cloud Analytic Services

Steven Sober, Brian Kinnebrew, SAS Institute Inc.

ABSTRACT

This is an introductory paper to help you understand how to convert SAS® code to leverage the distributed computing engine in SAS® Cloud Analytic Services (CAS). As customers adopt SAS® Viya®, they have access to many new visual interfaces, procedures, and CAS action sets that execute code in CAS. But what about existing SAS routines? How does one convert these routines to leverage CAS? In this session, we will cover terminology, the sweet spot in CAS, the biggest SAS®9 gotchas in the conversion of SAS code to SAS Viya and CAS, best practices in doing the conversion, workarounds, case studies, coding examples, and reading material. Additionally, you will learn how to engage with SAS to run your SAS code through the SAS Viya readiness assessment utility, which produces reports to assist you in your conversion of SAS code to leverage CAS.

INTRODUCTION

SAS Viya has two compute engines.

The SAS Programming Runtime Environment (SPRE) is also referred to as the compute server in SAS Viya. You can think of this engine as a SAS Workspace Server. The primary difference between SPRE and a workspace server is SPRE does not support the SAS® Metadata Server technology. This is due to SAS Viya moving to Micro Services application whereas the SAS Metadata Server is a monolithic application.

The second compute engine is CAS. The CAS engine is an in-memory engine. This means that the data is in memory and spread across all threads on all CAS worker nodes. The processing of this in-memory data is distributed across all the threads and happens in parallel. This can reduce run times of CAS enabled procedures and DATA Steps.

It is important to realize not all procedures and DATA Step statements are CAS enabled. This is because not all procedures and statements are eligible for distributed processing. These procedures and statements will simply run in SPRE (the compute server in SAS Viya).

See these resources to view a list of SAS Viya procedures by product name:

- Base SAS
- SAS/CONNECT®
- SAS® Econometrics
- SAS® IML
- SAS® Optimization
- SAS/QC® software
- SAS/STAT® software
- SAS® Scalable Performance Data Server
- SAS® Visual Data Mining and Machine Learning
- SAS® Visual Forecasting

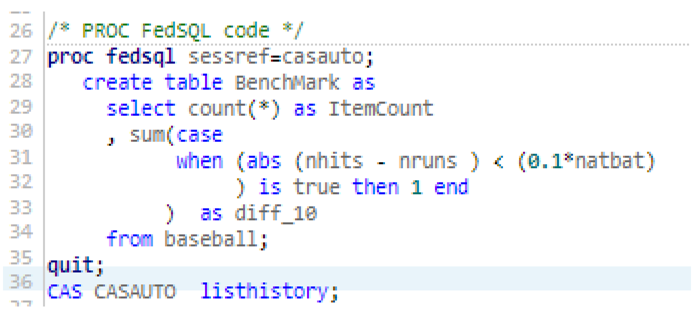
- SAS® Visual Statistics
- SAS® Visual Text Analytics

THE SWEET SPOT FOR CAS

There are multiple sweet spots for CAS:

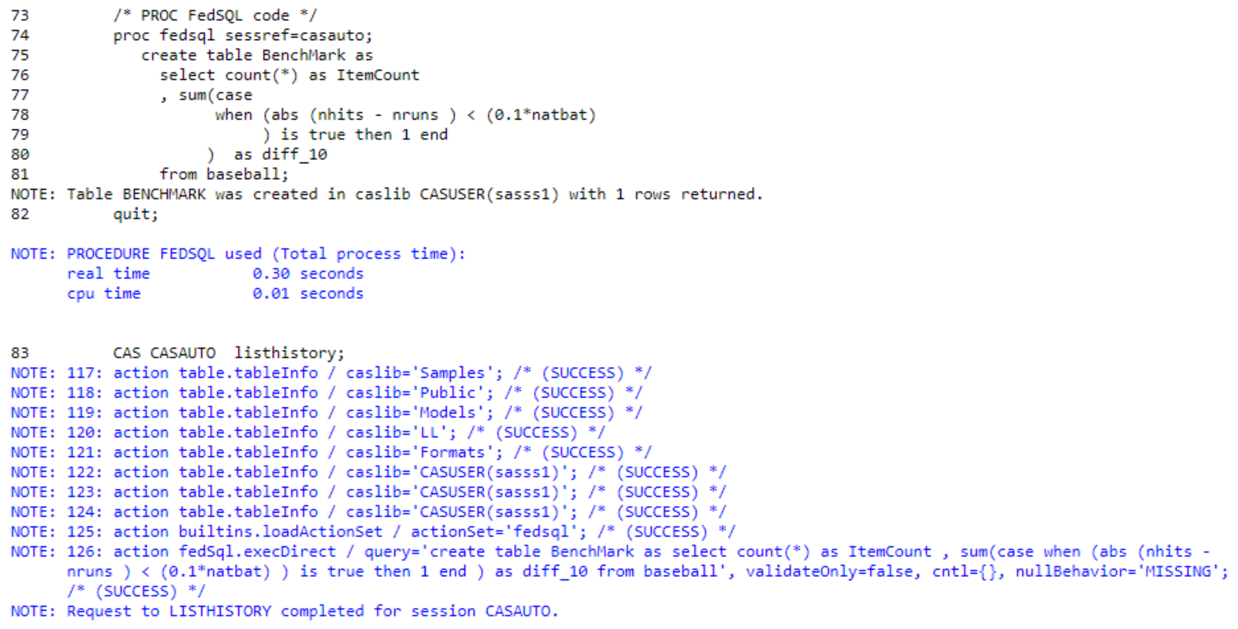
- SAS Visual Analytics
- SAS Visual Data Mining and Machine Learning
- SAS Visual Forecasting
- SAS Visual Statistics
- SAS Artificial intelligence (AI)
- SAS Analytical Procedures
    - SAS®9 analytical procedures need to be converted to their corresponding CAS enabled procedures. For example PROC LOGISTIC runs in SPRE. Consider converting it to PROC LOGSELECT, which runs in CAS.
- SAS IoT
- SAS® Cloud Analytics Services Language (CASL)
    - CASL is a new coding component in SAS Viya. A benefit of CASL is that you can easily convert it into open source code, such as Python and R. As a result, you can use the coding technique that you are most comfortable with and collaborate with other team members to meet business objects.
- For SAS programmers, one way to learn CASL is by reviewing the SAS log. To understand the CASL code generated by a CAS enabled procedure and or DATA Step, add the following statement (see line 36 in Figure 1) after the step:

```
CAS <CAS SESSION NAME> listhistory;
```



```
26 /* PROC FedSQL code */
27 proc fedsql sessref=casauto;
28    create table BenchMark as
29      select count(*) as ItemCount
30      , sum(case
31            when (abs (nhits - nruns ) < (0.1*natbat)
32                 ) is true then 1 end
33           )  as diff_10
34      from baseball;
35 quit;
36 CAS CASAUTO  listhistory;
```

Figure 1. SAS Code to Generate CASL Code in the SAS Log



```
73         /* PROC FedSQL code */
74         proc fedsql sessref=casauto;
75            create table BenchMark as
76              select count(*) as ItemCount
77              , sum(case
78                    when (abs (nhits - nruns ) < (0.1*natbat)
79                         ) is true then 1 end
80                   )  as diff_10
81              from baseball;
NOTE: Table BENCHMARK was created in caslib CASUSER(sasss1) with 1 rows returned.
82         quit;

NOTE: PROCEDURE FEDSQL used (Total process time):
      real time           0.30 seconds
      cpu time            0.01 seconds


83         CAS CASAUTO  listhistory;
NOTE: 117: action table.tableInfo / caslib='Samples'; /* (SUCCESS) */
NOTE: 118: action table.tableInfo / caslib='Public'; /* (SUCCESS) */
NOTE: 119: action table.tableInfo / caslib='Models'; /* (SUCCESS) */
NOTE: 120: action table.tableInfo / caslib='LL'; /* (SUCCESS) */
NOTE: 121: action table.tableInfo / caslib='Formats'; /* (SUCCESS) */
NOTE: 122: action table.tableInfo / caslib='CASUSER(sasss1)'; /* (SUCCESS) */
NOTE: 123: action table.tableInfo / caslib='CASUSER(sasss1)'; /* (SUCCESS) */
NOTE: 124: action table.tableInfo / caslib='CASUSER(sasss1)'; /* (SUCCESS) */
NOTE: 125: action builtins.loadActionSet / actionSet='fedsql'; /* (SUCCESS) */
NOTE: 126: action fedSql.execDirect / query='create table BenchMark as select count(*) as ItemCount , sum(case when (abs (nhits -
      nruns ) < (0.1*natbat) ) is true then 1 end ) as diff_10 from baseball', validateOnly=false, cntl={}, nullBehavior='MISSING';
      /* (SUCCESS) */
NOTE: Request to LISTHISTORY completed for session CASAUTO.
```

Figure 2. SAS Log Containing the Generated CASL Code

SAS Base procedures and DATA Steps are not considered a sweet spot for CAS. This is due to some SAS Base procedures (such as PROC MEANS and PROC SUMMARY) splitting the processing of the CAS table between CAS and SPRE. With DATA Steps, data volumes matter. For a DATA Step to perform in CAS, your CAS source tables need to be 50GB or larger. For data volumes less than 50GB, simply run that DATA Step in SPRE. In addition to data volumes, you want to have DATA Steps that are CPU intensive. By CPU intensive we mean the CPU Time reported in the SAS log is equal to or larger than the real time reported in the SAS log.

Lastly, not all things need to run in CAS to justify SAS Viya. SAS Viya is the future of SAS and adoption of it is necessary to stay in sync with SAS.

THE BIGGEST SAS®9 GOTCHAS IN CONVERTING SAS CODE TO SAS VIYA

SAS®9 HABITS AND PRODUCTS

SAS programmers have the habit of not cleaning up data sources that are not needed in subsequent steps. With in-memory processing, we must clean up to avoid our in-memory tables from being paged out to disk. Paging tables to disk and then back into memory adds latency to our real times that are reported in the SAS Log. In our GitHub repository, there are coding examples on how to delete CAS tables.

For the SAS® Enterprise BI application (which includes stored processes, information maps and OLAP cubes), we advise using a SAS 9.4M5 or later client to access SAS Viya.

For SAS/IntrNet® software, PROC OLAP, PROC LASR, SAS/AF® software, SAS/FSP® software, SAS/SHARE® software, and other products with no plans to be supported in SAS Viya, we advise using a SAS 9.4M5 or later client to access SAS Viya.

For SAS® Enterprise Guide®, you have two options. The first is to continue to use SAS 9.4M5 or later as your client to access SAS Viya. The second option is to use SAS Enterprise Guide 8.2 with SAS Viya 3.5. This option removes the dependency of a SAS 9.4M5 (or later) client. Instead, SAS Enterprise Guide 8.2 submits code directly to SAS Viya. To use the second option, work with your SAS team to obtain permission from SAS to leverage SAS Enterprise Guide 8.2.

CONVERTING EVERY STEP TO LEVERAGE CAS

Do not do this. Instead, be selective in identifying steps that can leverage CAS.

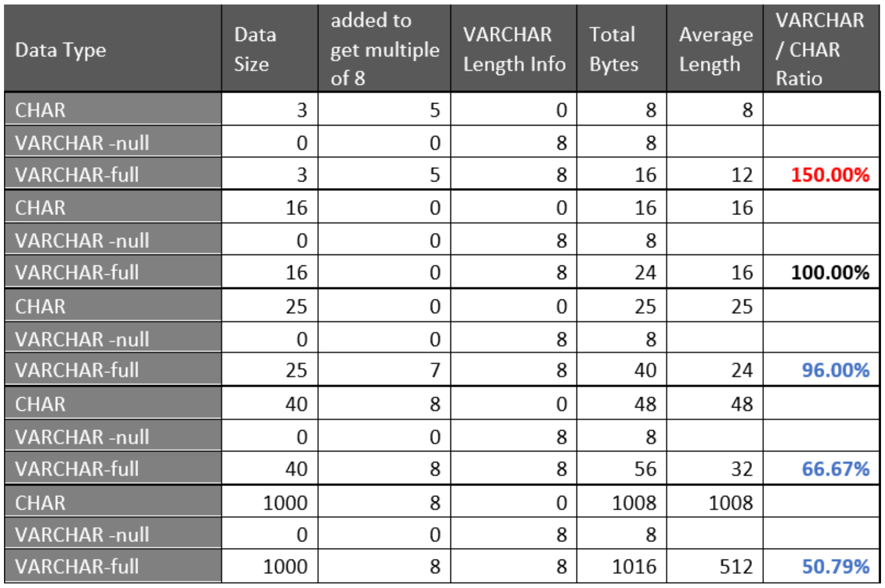
When converting, look for these steps:

- Long-running steps, which are steps that run for 30 minutes to multiple hours.
- CPU intensive steps.
  - You must consider the computational demand of your code when determining where to execute your programs.
  - The real time for the step is equal to or less than the combined CPU time.
  - Convert analytical procedures that are computationally demanding (such as PROC LOGISTIC, PROC MIXED, or PROC REG) to their CAS procedure. Here are some examples:
    - PROC LOGISTIC (SPRE) ➔ PROC LOGSELECT (CAS)
    - PROC MIXED (SPRE) ➔ PROC LMIXED (CAS)
    - PROC REG (SPRE) ➔ PROC REGSELECT (CAS)
- DATA Steps containing a lot of computations, functions, and conditional logic. An example is a Monte Carlo Simulation with thousands of lines of DATA Step code.
- Data volumes greater than 50GB. This is a general baseline recommended by SAS.
- Cardinality of BY variables.
  - With high cardinality, there are just a few records with each distinct value (for example, at the loan level and at the patient level).
  - DATA Step BY variables might run faster SPRE.
  - An exception is when you generate a partition CAS table and the BY statement matches the partition variables. In the GitHub repository, we have an example of creating a partitioned CAS table.
- Data order of CAS tables.
  - There is no concept of ascending or descending with symmetric multiprocessing (SMP) or massively parallel processing (MPP) distributed processing.
  - BY statements group the data correctly across the CAS worker nodes and threads.
- Sampling data because you had to with SAS®9. With CAS, consider running your code against the entire table.
- Moving data between CAS (CAS table) and SPRE (SAS7BDAT data set)
  - Use PROC CASUTIL to accomplish this. Note: This procedure requires a CASLIB CASLIB statements support parallel read of SAS7BDAT data sets when the shared file system has been mounted on all CAS worker nodes and all CAS worker nodes have direct access to the physical path of the SAS7BDAT data sets.

- For distributed computing (CAS), there is a cost that is incurred.
  - There is latency with services having to start up to run the code distributed.

    - BY statements are done on the fly, which means the CAS table is re-distributed across the CAS worker nodes and threads. The exception to this is when you create a partition CAS table and the BY statement in a given CAS enabled step matches the partition.

- Data volumes increase when lifting the data into CAS.
  - CAS data is closely coupled with its memory representation and requires alignment to 8-byte boundaries for memory mapping.
  - All variable lengths must be a multiple of 8, which means record lengths and data size will increase. Here are some examples:
    - Six variables are $CHAR3. or 18 bytes total. In CAS, each variable is increased by 5 bytes and totaled 48 bytes.
    - One variable is $CHAR9. or 9 bytes total. In CAS, the variable increased by 7 bytes and totaled 16 bytes.
    - One variable is numeric with a length of 8. In CAS, there is no increase in size.
    - The record length increased from 35 bytes to 72 bytes.
  - VARCHAR
    - For character variables with lengths less than 16, keep them as $CHARw.
    - For character variables with lengths greater than or equal to 16, make them VARCHAR(w).
    - Figure 3 compares the ratio of space needs for the VARCHAR and CHARACTER data types.



Data Type	Data Size	added to get multiple of 8	VARCHAR Length Info	Total Bytes	Average Length	VARCHAR / CHAR Ratio
CHAR	3	5	0	8	8	
VARCHAR -null	0	0	8	8		
VARCHAR-full	3	5	8	16	12	**150.00%**
CHAR	16	0	0	16	16	
VARCHAR -null	0	0	8	8		
VARCHAR-full	16	0	8	24	16	**100.00%**
CHAR	25	0	0	25	25	
VARCHAR -null	0	0	8	8		
VARCHAR-full	25	7	8	40	24	**96.00%**
CHAR	40	8	0	48	48	
VARCHAR -null	0	0	8	8		
VARCHAR-full	40	8	8	56	32	**66.67%**
CHAR	1000	8	0	1008	1008	
VARCHAR -null	0	0	8	8		
VARCHAR-full	1000	8	8	1016	512	**50.79%**

Figure 3. Space Needs for VARCHAR and CHARACTER Data Types

- Small data sets / tables (less than 50GB)
  - Keep these tables as SAS7BDAT and run your code in SPRE.
  - For CAS enabled analytical procedures (such as PROC LOGSELECT), these small tables will automatically get lifted into CAS for processing.

    - For CASL, you must use PROC CASUTILT to lift the SAS7BDAT data set to a CAS table.
- DATA Step
    - In SMP mode (one CAS worker node), the DATA Step might perform better in SPRE.
        - When executing a DATA Step in SPRE, ensure the source and target tables to those DATA Steps are SAS7BDAT data sets and not CAS tables.
        - Here are two exceptions:
            - when you generate a partition CAS table and the BY statement matches the partition variables
            - when you run a benchmark and prove that DATA Step is faster in CAS.
    - MPP mode (more than one CAS worker node)
        - Run a benchmark to determine whether to run that DATA step in CAS or SPRE.
        - An exception is when you generate a CAS table that is partitioned by BY group and the BY statement matches the partition BY group.
        - LAG, DIF, and RETAIN variables. If you initial all LAG, DIF and RETAIN variables using FIRST. processing, you can run the DATA Step distributed in CAS. If you are not doing this, you need to run your DATA Step in single threaded mode in SPRE.
    - _N_. When running distributed in CAS, _N_ will have the same value for each thread defined to your CAS environment. If you have 100 threads, you will have the value of 1, then 2, to *n* for each thread processing the DATA Step. As a result, 100 records have the value of 1, 100 records will have the value of 2, and so on. You will see different results when compared to running this DATA Step with a single thread in SPRE and or SAS®9.
- PROC FedSQL
    - PROC FedSQL is ANSI 99 SQL compliant. Databases that are ANSI 99 SQL compliant means that SQL can be pushed down to the database even though it is not that database's native SQL constructs.
    - PROC SQL is ANSI 92 SQL compliant with unique SAS constructs that are not available in FedSQL.
    - FedSQL running in CAS has consistently shown improved runtimes in SMP mode and MPP mode and with data volumes less than 50GB.
- By default, the SAS log reports only the real time, CPU-time, and memory usage for SPRE. To review detailed CAS performance metrics in the SAS log, you must use the SESSOPTS option METRICS=TRUE.
- Finally, we strongly recommend you request a SAS Viya sizing before staging hardware. To request a sizing, work with your SAS team.

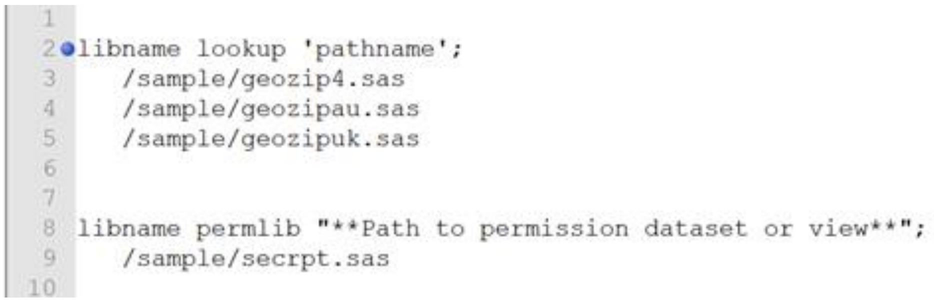
STEP ONE, TWO, THREE

There are three primary steps you should follow to ensure a successful migration of your code to CAS.

STEP ONE

Test your programs in SPRE. Make any necessary modifications so your code executes successfully. Here are some things to remember:

- Authentication domains are not supported in LIBNAME statements in SPRE.
- Metadata LIBNAME statements are not supported in SPRE.
- The SASIOLA (SAS® LASR™ Analytic Server) engine is not supported in SPRE.
- For additional information on LIBNAMEs in SAS®9, review the libnames.txt report that is produced by the SAS Viya readiness assessment utility.



```

libname lookup 'pathname';
   /sample/geozip4.sas
   /sample/geozipau.sas
   /sample/geozipuk.sas


libname permlib "**Path to permission dataset or view**";
   /sample/secrpt.sas

```

Figure 1. Sample Output from the libnames.txt Report

STEP TWO

Create CASLIBs for all data sources, including shared file systems, relational databases (RDBMSs), and cloud data repositories (such as AWS S3 buckets). A caslib is necessary so that you can connect to your various data sources when loading data into CAS or saving data from CAS back to the data source.

> Note: To establish an authentication domain within a CASLIB, use the addCASlib CAS action.

STEP THREE

Start with the step of a program that requires the greatest improvement in execution time. This might or might not be the first step or program in your overall process flow. Leverage the CAS engine as much as possible. This will provide you with the greatest opportunity for improvement. Also, leverage the "Detail reports" produced by the SAS Viya readiness assessment utility. These reports will serve as your roadmap and guide in knowing what code changes are necessary to leverage CAS.

WORKAROUNDS

In some instances, you might encounter a feature or capability that isn't yet supported in SAS Viya. Or, you might be able to leverage features and capabilities that are available in CAS that SAS®9 doesn't provide. In either case, we have developed several alternative approaches and best practices when migrating your code from SAS®9 to CAS. Examples of each can be found on our GitHub page.

- In certain situations, PROC SORT can be commented out or removed. In CAS, all BY statement processing is done during the execution of the code block that contains the BY statement. One exception is when the data has already been partitioned and ordered by the variables in the BY statement. In this case, no BY statement processing is necessary.

- Running a DATA Step with BY variables having high cardinality. In this situation, execution on SPRE might be faster. Operations on variables with high cardinality often do not perform well in distributed computing environments. Test the code in both CAS and SPRE to determine the optimal run-time engine.
- Running a DATA Step with the DESCENDING option on numeric variables. In SAS Viya 3.4 or earlier, the DESCENDING option is not supported in CAS. Alternatively, you can create a CAS view and calculate new variables that contain the negative value of the original numeric BY variables. Use these new variables in the BY statement in ascending order. This will result in the original numeric BY variables ordered in a descending fashion.

  In SAS Viya 3.5, DATA Steps containing the DESCENDING option in the BY statement are supported for numeric, character, and varchar data types. However, this only applies to all but the first BY variable. If there is only one BY variable, the DESCENDING option is not supported.
- In SAS Viya 3.5, the NODUPKEY=, NOUNIKEY=, DUPOUT=, and UNIOUT= options have been added to PROC SORT. If using the NODUPKEY= or NOUNIKEY= options and writing the output to CAS via the DUPOUT= or UNIOUT= options, PROC SORT invokes the new deduplicate CAS action. In this case, all work is performed on the CAS server.

  In SAS Viya 3.4 and earlier, a work-around for deduplication (PROC SORT with the NODUPKEY= option) must be employed. There are a few different ways this can be accomplished.
  - o Emulate using a DATA Step. Output the observation containing the first occurrence of the last BY variable. Read the "How to achieve repeatable results by distributed DATA Step BY Groups" blog for more information. In distributed computing, the order of rows within a given BY group is random.
  - o Emulate with PROC FEDSQL – use DISTINCT and GROUP BY statements.
  - o Use the GroupByInfo CAS action.
- DATA Steps that use the SYMGET function are not supported in CAS. You can emulate these DATA Steps by substituting a format for the SYMGET function and applying the format with a PUT statement. You can also use the CAS Language (CASL) since SYMGET is one of many CASL built-in functions.
- PROC APPEND is one of the widely used SAS®9 procedures that is not yet CAS enabled. However, this is easily replicated by using a DATA Step with multiple tables in the SET statement and placing the DATA Step option (APPEND=YES or FORCE) on the target table. Use of this option requires SAS Viya 3.4.

CASE STUDIES

U.S. FINANCIAL COMPANY

This financial company leverages a Monte Carlo simulation to compute the propensity of delinquency. This Monte Carlo simulation is a DATA Step that contains over 15,000 lines of code. The source table is about 1TB, and the simulation's target table is about 50GB. The simulation runs multiple times and takes 1 hour per iteration with SAS®9. To convert this DATA Step to run distributed in CAS, we had to ensure the source and target table to the DATA Step were CAS tables, and we had to change seven occurrences of the RANUNI function. Once we changed the seven RANUNI function to the CAS enabled RAND function with the uniform parameter, our run time dropped from 1 hour to 6 minutes per iteration.

The overall run time of the simulation dropped from 26 hours in SAS®9 to 2 and ½ hours in using the CAS engine.

The business benefits by leveraging the CAS engine are:

- Quicker time to offer.
- The window where no other processes can run while the simulation is running on SAS Viya is reduced to 2 and ½ hours. With SAS®9 that window was 26 hours.

U.S. GOVERNMENT

A branch of the U.S. Government had a process whereby they were running 51 iterations of a program to create analytical base tables for modeling and scoring. This process was comprised of DATA Steps and Base SAS procedures and ran on their existing SAS 9.4 platform. The source data ranged in size from 220GB – 230GB. The target analytical base tables ranged in size from 44GB – 46GB.

To execute this process in CAS, here is what they did based on recommendations from SAS:

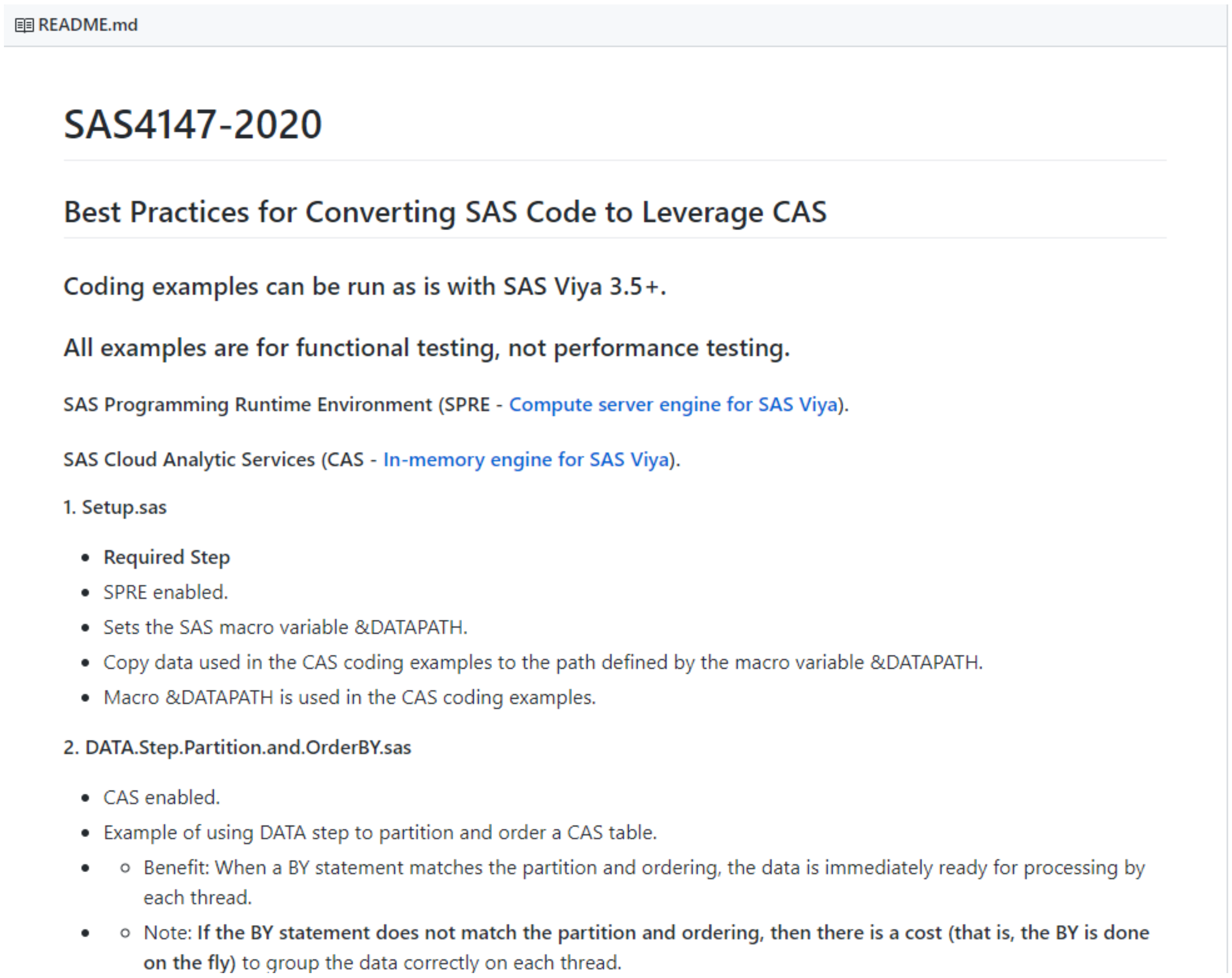
- Changed all source and target tables to utilize CASLIBs instead of SAS 9.4 LIBREFs.
- Ran code blocks with BY statements containing variables with high cardinality (~50MM unique values) in SPRE. Through a series of tests and comparisons, it was determined that SPRE is more effective at handling BY statement processing of high cardinality variables. Here is the process:
    - Utilized PROC SORT to prepare source data for merging. Source tables resided in CAS and target tables were output to WORK in SPRE.
    - Performed a DATA Step MERGE with source and target tables residing in WORK in SPRE.
    - Executed PROC CASUTIL to load merged WORK data set into CAS.
- Converted all SQL procedures to FedSQL procedures to leverage CAS. PROC SQL is not CAS enabled.
- Converted all FREQ procedures to FREQTAB procedures to leverage CAS. PROC FREQ is not CAS enabled although the freq CAS action could also have been used.
- Converted LOGISTIC procedure to LOGSELECT procedure to leverage CAS. PROC LOGSELECT is the SAS Viya replacement for PROC LOGISTIC.
- Added a DATA Step to call the score code produced by PROC LOGSELECT using a %include statement.

The result was an overall reduction in execution time from 56 hours in SAS 9.4 to 9 hours in CAS. The CAS enabled DATA Steps and Base SAS procedures took 7 hours in CAS as opposed to 18 hours in SAS 9.4. The time to execute the logistic regression fell from 38 hours in SAS 9.4 using PROC LOGISTIC to 2 hours in CAS using PROC LOGSELECT. This faster time to results allowed the customer to increase their frequency of process execution, all the while maintaining the accuracy of the results between SAS 9.4 and SAS Viya.

GITHUB – CODING EXAMPLES

To help you get started in writing code to leverage CAS, we have created a GitHub repository. Simply submit "1.Setup.sas", which defines the macro variable &DATAPATH that points to a path that you have access to in your SAS Viya environment. &DATAPATH is referenced in all other coding examples. In addition, this step copies the SASHELP.BASEBALL and SASHELP.CARS to the path specified by &DATAPATH.

The examples are designed for functional testing to better understand how to write SAS code to leverage CAS. These examples are not designed to show performance gains by leveraging CAS.


README.md

SAS4147-2020

Best Practices for Converting SAS Code to Leverage CAS

Coding examples can be run as is with SAS Viya 3.5+.

All examples are for functional testing, not performance testing.

SAS Programming Runtime Environment (SPRE - Compute server engine for SAS Viya).

SAS Cloud Analytic Services (CAS - In-memory engine for SAS Viya).

1. Setup.sas

- **Required Step**
- SPRE enabled.
- Sets the SAS macro variable &DATAPATH.
- Copy data used in the CAS coding examples to the path defined by the macro variable &DATAPATH.
- Macro &DATAPATH is used in the CAS coding examples.

2. DATA.Step.Partition.and.OrderBY.sas

- CAS enabled.
- Example of using DATA step to partition and order a CAS table.
-
  - Benefit: When a BY statement matches the partition and ordering, the data is immediately ready for processing by each thread.
-
  - Note: **If the BY statement does not match the partition and ordering, then there is a cost (that is, the BY is done on the fly)** to group the data correctly on each thread.

Figure 5. GitHub Repository

CONCLUSION

The CAS engine in SAS Viya is a powerful distributed computing engine that can drastically reduce run times of SAS processes. The thoughts laid out in this paper will help you in deciding what makes sense to CAS enable.

RECOMMENDED READING

- Blogs by Steven Sober
- Blogs by Brian Kinnebrew
- SAS Community Articles by Steven Sober
- SAS Community Articles by Brian Kinnebrew
- "How to achieve repeatable results with distributed DATA Step BY Groups" in blog post by Steven Sober
- "Six reasons you should stop using the RANUNI function to generate random numbers" in blog post by Rick Wicklin

- Building Machine Learning Models by Integrating Python and SAS Viya by Kris Stobbe
- SAS 9.4 and SAS Viya 3.5 Programming Documentation
- SAS Cloud Analytic Services
- Parallel Programming with the DATA Step: Next Steps, SAS Global Forum paper by David Bultman and Jason Secosky
- Base SAS Procedures That Use CAS actions
- *SAS Viya 3.5: FedSQL Programming for SAS Cloud Analytic Services*
- FedSQL Implicit Pass-Through Facility in CAS in *SAS Viya 3.5: FedSQL Programming for SAS Cloud Analytic Services*

CONTACT INFORMATION

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

Steven Sober
SAS Institute Inc.
919-531-9644
steven.sober@sas.com

Brian Kinnebrew
SAS Institute Inc.
469-801-7461
brian.kinnebrew@sas.com