Best Practices for Migrating SAS Code to Leverage CAS

Steven Sober - Advisory Solutions Architect
Brian Kinnebrew - Principal Solutions Architect
26Aug2019
Best Practices for Migrating SAS Code to Leverage CAS

Agenda

• Terminology (Steven)
• CAS’s Sweet Spot (Steven)
• The Biggest SAS 9 Gotchas (issues) in Migration of SAS code to SAS Viya are…. (Steven)
• Steps One Two Three (Brian)
• Workarounds (Brian)
• Case Studies (Steven and Brian)
• Appendix – Coding Examples (Steven and Brian)
• Reading Materials (You)
• Next Steps (Steven and Brian)
Terminology
Best Practices for Migrating SAS Code to Leverage CAS

Terminology

• SAS Viya has two compute engines
  • SAS Programming Runtime Environment (SPRE)
    • Also referred to as SAS Viya’s Compute Server
    • Also referred to as SAS Viya’s Workspace Server
  • CAS (pronounced CAZ)
    • The in-memory engine
      • Data is distributed across all CAS worker nodes and threads
      • Processing is done in parallel
    • Not all SAS Viya Procedures are CAS enabled
    • Not all DATA Step statements are CAS enabled

• SAS Viya Procedures
  • SAS Products
  • Base SAS
  • SAS/CONNECT
  • SAS Cloud Analytic Services
  • SAS Econometrics
  • SAS/IML
  • SAS Optimization
  • SAS/QC
  • SAS Scalable Performance Data Server
  • SAS Visual Data Mining and Machine Learning
  • SAS Visual Forecasting
  • SAS Visual Statistics
  • SAS Visual Text Analytics
CAS’s Sweet Spot
Best Practices for Migrating SAS Code to Leverage CAS

CAS’s Sweet Spot

• SAS Viya’s Visual interfaces (Sweet Spot)

• CAS enabled analytical procedures and CAS action sets/actions (Sweet Spot)
  • SAS 9 analytical procedures need to be re-coded to be CAS aware
  • Review the frequency report produced by the SAS Viya Readiness Assessment Utility for hints on refactoring a procedure to be CAS enabled

<table>
<thead>
<tr>
<th>Candidate for CAS by Converting to PROC LMMIXED - PROC MIXED</th>
<th>42</th>
<th>0.35</th>
<th>5691</th>
<th>47.68</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate for CAS by Converting to PROC LOGSELECT - PROC LOGISTIC</td>
<td>85</td>
<td>0.71</td>
<td>5776</td>
<td>48.39</td>
</tr>
</tbody>
</table>

• Open Source integration (Sweet Spot)

• **Data volume greater than 50GB** (Sweet Spot)
  • Meaning the SAS7BDAT data set and or RDBMS table you want to lift into CAS
  • This is a general baseline recommended by R&D
  • You must also consider the computational demand of your code when determining where to execute your programs
Best Practices for Migrating SAS Code to Leverage CAS

CAS’s Sweet Spot

• Base SAS routines (Sweet Spot, NOT)
  • **CAS enabled Base SAS procedures** and DATA Step can leverage CAS but **one must benchmark to determine which engine is the fastest** (i.e. SPRE – SAS Viya’s Compute Server or CAS).
    • May perform best in SPRE (SAS Viya’s Compute Server)
    • Some CAS enabled Base SAS procedures execute partly in SPRE (SAS Viya’s Compute Server).
    • Additionally, some statements and functions are not supported in CAS.
      • In these cases, use CASL for 100% execution in CAS.

• **Remember to clean up CAS tables as soon as subsequent job steps do not need them**
  • SAS programmers have a bad habit of never deleting temporary tables (SASWORK data sets) once they are done with them
  • With CAS if you do not clean up as you go you will start hitting CAS Disk Cache which means you will be causing unnecessary I/O
  • In the Appendix we have coding examples on how to delete CAS tables

• **All things do not need to run in CAS to justify SAS Viya!**
  • **Review first 4 bullet points in this section (CAS’s Sweet Spot)**
The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are....
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

• **Enterprise BI application**
  
  • **Stored Processes, Information Maps, OLAP Cubes**
    - We advise to use a SAS9.4M5+ client to SAS Viya
    - Stored processes are planned for migrate to SAS Viya as jobs
      - The goal is to convert them automatically, including complex/cascading prompts
    - Automated translation of InfoMaps and OLAP Cubes (but not Proc OLAP code) are planned to migrate to SAS Viya as jobs

• **SAS/IntrNet, PROC OLAP, PROC LASR**
  
  • No plans to support these in SAS Viya
  
  • We advise to use a SAS 9.4M5+ client to SAS Viya
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

- **EG projects**
  - We advise to use a SAS 9.4M5+ client to SAS Viya
  - With EG 7.15 (SAS 9.4M5 shipped 2017 week 38) CAS enabled tasks were added

- **SAS/Share, SAS/AF, SAS/FSP and other products with no plans to be support in SAS Viya**
  - We advise to use a SAS 9.4M5+ client to SAS Viya
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

• Converting every step to leverage CAS
  • Do not do this!
  • Instead:
    • Be selective in identifying steps that can leverage CAS, look for:
      • Long running steps – 30 minutes to multiple hours
      • CPU intensive steps
        • Analytical procedures that are computationally demanding like a REGRESSION, MIXED or LOGISTIC
        • The real time for the step is equal to or less than the combined CPU time
      • DATA Step containing a lot of computations, functions, and conditional logic
        • For example, a Monte Carlo Simulation with thousands of lines of DATA Step code
      • Data volumes greater than 50GB
        • This is a general baseline recommended by R&D.
        • You must also consider the computational demand of your code when determining where to execute your programs.
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

• Moving data between CAS (CAS Table) and SPRE – SAS Viya’s Compute Server (SAS7BDAT)
  • Use PROC CASUTIL to accomplish this
    • This procedures require a CASLIB
    • CASLIB statements support parallel read of SAS7BDAT datasets 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 datasets

• Sampling data because you had to with SAS 9
  • With CAS, consider running your code against the entire table
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

• Small data (< 50GB)
  • Keep these tables as SAS7BDAT and run your code in SPRE (SAS Viya’s Compute Server)
  • This is a general baseline recommended by R&D
  • You must also consider the computational demand of your code when determining where to execute your programs

• Cardinality of BY variables
  • With high cardinality, there are just a few records with each distinct value (i.e. at the loan level, at the patient level)
  • May run fastest SPRE
  • **Exception is when you generate a partition CAS table and the BY statement matches the partition variables** – In the appendix we have a coding example of creating a partitioned CAS Table

• Data order of CAS tables
  • BY statements group the data correctly across the CAS Worker nodes and threads
    • **However there is no concept of Ascending or Descending with SMP or MPP distributed processing**
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

• 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
    • Data 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 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 – for example:
    - Six variables are $CHAR3. or 18 bytes total, Increasing each variable by 5 bytes equals 48 bytes total
    - One variable is $CHAR9. or 9 bytes total, increasing by 7 bytes equals 16 bytes total
    - One variable is numeric with a length of 8, no increase in size
    - The record length increased from 35 bytes to 72 bytes
Best Practices for Migrating SAS Code to Leverage CAS
The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

- **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)

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Data Size</th>
<th>added to get multiple of 8</th>
<th>VARCHAR Length Info</th>
<th>Total Bytes</th>
<th>Average Length</th>
<th>VARCHAR / CHAR Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>VARCHAR-null</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR-full</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>16</td>
<td>12</td>
<td>150.00%</td>
</tr>
<tr>
<td>CHAR</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>VARCHAR-null</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR-full</td>
<td>16</td>
<td>0</td>
<td>8</td>
<td>24</td>
<td>16</td>
<td>100.00%</td>
</tr>
<tr>
<td>CHAR</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>VARCHAR-null</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR-full</td>
<td>25</td>
<td>7</td>
<td>8</td>
<td>40</td>
<td>24</td>
<td>96.00%</td>
</tr>
<tr>
<td>CHAR</td>
<td>40</td>
<td>8</td>
<td>0</td>
<td>48</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>VARCHAR-null</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR-full</td>
<td>40</td>
<td>8</td>
<td>8</td>
<td>56</td>
<td>32</td>
<td>66.67%</td>
</tr>
<tr>
<td>CHAR</td>
<td>1000</td>
<td>8</td>
<td>0</td>
<td>1008</td>
<td>1008</td>
<td></td>
</tr>
<tr>
<td>VARCHAR-null</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR-full</td>
<td>1000</td>
<td>8</td>
<td>8</td>
<td>1016</td>
<td>512</td>
<td>50.79%</td>
</tr>
</tbody>
</table>
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

- **DATA Step**
  - **SMP mode (one CAS worker node)**
    - DATA Step in SMP mode may perform better in SPRE (SAS Viya Compute Server)
    - When executing DATA Step in SPRE ensure the source and target tables to those DATA Steps are SAS7BDAT datasets and not CAS tables
    - Exception is when you generate a partition CAS table and the BY statement matches the partition variables
    - Exception is when one runs a benchmark and proves DATA Step is faster in CAS
  - **MPP mode (more than one CAS worker node)**
    - Needs to be benchmarked to determine where to run that DATA Step (i.e. CAS or SPRE – SAS Viya’s Compute Server)
    - Exception is when you generate a partition BY group CAS table and the BY statement matches the partition BY group

- **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
Best Practices for Migrating SAS Code to Leverage CAS

The Biggest SAS 9 Gotchas (issues) in Migration of SAS Code to SAS Viya are...

- For Distributed Computing (CAS) there is a cost that is incurred
  - By default the SAS log only reports SPRE’s (SAS Viya’s Compute Server) Real-time, CPU-time and Memory usage
  - To review detailed CAS performance metrics in the SAS log you must use the sessopts option METRICS=TRUE
- Request a SAS Viya sizing be completed before staging hardware
  - You can request the sizing through your SAS account executive
Steps One Two Three
Best Practices for Migrating SAS Code to Leverage CAS
Steps One Two Three

• Step one
  • Run your programs successfully in SPRE (SAS Viya’s Compute Server) using SAS9 LIBNAMEs
    • Authentication domains are not supported
    • Metadata LIBNAME Statement are not supported
    • SASIOLA engine is not supported
  • For information on SAS 9 LIBNAMEs 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/secret.sas
```
Best Practices for Migrating SAS Code to Leverage CAS

Steps One Two Three

• Step two
  • Create CASLIBs for all data sources (i.e. SAS7BDAT, RDBMS’s)
    • CASLIB authentication domains are supported by leveraging the `CAS table.addCaslib` action

• Step three
  • **Starting with the most important step**, not job, **to speed up**
    • Leverage CAS as much as possible
    • Leverage the “**Detail reports**” that are produced by the SAS Viya Readiness Assessment Utility
      • These reports are your roadmap in knowing what to change to leverage CAS
Best Practices for Migrating SAS Code to Leverage CAS

Steps One Two Three
Workarounds
Best Practices for Migrating SAS Code to Leverage CAS

Workarounds

• PROC SORT code is a candidate for commenting out
  • All BY statements that run in CAS are done on the fly
    • There is one exception which is when the BY statement matches the partitioning and order by variables of a partitioned CAS table
      • See appendix for an example of using DATA Step to partition and order by columns of a given CAS table

• DATA Step BY variables with high cardinality
  • You might have to run these in SPRE (SAS Viya’s Compute Server), see appendix for code

• DATA Step with DESCENDING BY on numeric variables, see appendix for code
  • Create a CAS view with new variables that are the negated value of the original numeric BY variables
  • Use the new variables on the BY (in ascending order)
    • The original numeric variable will now be in descending order

• PROC SORT NODUPKEY, see appendix for code
  • Emulate with DATA Step by outputting a record for the first occurrence of the last BY variable
  • Emulate with FedSQL using DISTINCT and GROUP BY statements
  • The CASL action GroupByInfo can be used for removing duplicates
Best Practices for Migrating SAS Code to Leverage CAS

Workarounds

• **DATA Step de-duping, read this blog**
  • In distributed computing the order of the rows of a given BY group is random

• **DATA Step with SYMGET**
  • Emulate these statements with FORMATS
  • Runs in SPRE (SAS Viya’s Compute Server):
    • /* A macro variable is used in the expression */
    • varxyz = (-1)* (symget(compress('varabc' || varxyz || varhij)))/24 * varstv;
  • Runs in CAS
    • /* A formatted value is used in the expression */
    • varxyz =(-1)* (put(compress(varabc '|| varxyz || varhij), $varfmt))/24 * varstv;

• **PROC APPEND, see appendix for code**
  • Emulate with DATA Step with multiple tables on SET and the data set option (APPEND=YES) on the target table
  • Requires SAS Viya 3.4
Case Studies
Best Practices for Migrating SAS Code to Leverage CAS

Case Study – DATA Step

• Financial Use Case
  • Monte Carlo Simulations for Delinquency
    • DATA Step
      • ~20,000 line
      • Source table ~1TB
      • Target table ~50GB
    • Multiple Iteration of DATA Step
      • SAS Workspace Server - 1 hour per iteration
  • Business Benefit
    • Quicker time to offer
    • No other processes were allowed to run while Monte Carlo simulation runs
      • 26 Hours (Workspace Server)
      • 2 ½ Hours (CAS)

• To CAS Enable DATA Step
  • Changed multiple RANUNI functions (SPRE – SAS Viya’s Compute Server) to RAND function (CAS Enabled)
  • Changed source and target tables in DATA Step to use CASLIB instead of 9.4 LIBREF
  • Commented out PROC APPEND
  • Created CAS enabled DATA Step to emulate append process
  • Multiple iteration of CAS enabled DATA Step
    • CAS - 6 minutes per iteration
Best Practices for Migrating SAS Code to Leverage CAS

Case Study – DATA Step – PROC LOGISTIC

• US Government Use Case
  • **51 Iterations** of program for Analytical Base Table creation for Modeling and Scoring
    • DATA Step and Base PROCS
  • Source tables: ~220GB – 230GB
  • Target tables: ~44GB – 46GB

• Business Benefit
  • Faster time to results which allows for increased frequency of process execution.
  • Maintaining accuracy of results between 9.4 and SAS Viya
    • Workspace server (SAS 9)
      • 56 hours
    • CAS
      • 9 hours

• To CAS Enable the Process - Little to no changes required!
  • Changed source and target tables in DATA Step to use CASLIB instead of 9.4 LIBREF
  • **Ran BY statements with high cardinality variables (~50 million unique ids) in SPRE (SAS Viya’s Compute Server)**
    • PROC SORT (SPRE, source CAS table, target WORK table)
    • DATA Step (SPRE, source WORK table, target WORK table)
    • PROC CASUTIL to lift WORK table into CAS
  • Changed PROC SQL to PROC FEDSQL for CAS execution
  • Changed PROC FREQ to PROC FREQTAB for CAS execution
  • Converted PROC LOGISTIC to PROC LOGSELECT for CAS execution.
  • Added subsequent DATA Step to call score code using %INCLUDE statement.
  • **CAS Enabled DATA Step and Base PROCS**
    • 18 hours -> 7 hours
  • **CAS Enabled PROC LOGISTIC (PROC LOGSELECT)**
    • 38 hours -> 2 hours
Appendix – Coding Examples
Best Practices for Migrating SAS Code to Leverage CAS
Appendix – How to Display SAS Viya Version and Environment

cas;
caslib _all_ assign;
%put &sysvlong4.;

cas casauto listabout;

proc cas;
   about;
run;
Best Practices for Migrating SAS Code to Leverage CAS
Appendix – How to Display SAS Viya Version and Environment

76 %put &sysvlong4.;
V.03.04M0P07112018

77 cas casauto listabout;
Section: About
CAS = Cloud Analytic Services
Version = 3.04
VersionLong = V.03.04M0P07122018
Copyright = Copyright © 2014-2018 SAS Institute Inc. All Rights Reserved.
ServerTime = 2019-05-16T17:16:30Z

Section: System
Hostname = gtp-SAS Viya-controller
OS Name = Linux
OS Family = LIN X64
OS Release = 3.10.0-862.9.1.el7.x86_64
OS Version = #1 SMP Wed Jun 27 04:30:39 EDT 2018
Model Number = x86_64
Linux Distribution = Red Hat Enterprise Linux Server release 7.5 (Maipo)

Section: license
site = gtp34 GA
siteNum = 70180938
expires = 05Aug2019:00:00:00
gracePeriod = 45
warningPeriod = 45
maxCPUs = 9999
# Best Practices for Migrating SAS Code to Leverage CAS

## Appendix – How to Display SAS Viya Version and Environment

### Results from `builtins.about`

<table>
<thead>
<tr>
<th>Node Count</th>
<th>Total Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node Name</th>
<th>Role</th>
<th>Uptime (Sec)</th>
<th>Running</th>
<th>Stalled</th>
</tr>
</thead>
<tbody>
<tr>
<td>gtp-viya-worker-1.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>24.999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-worker-2.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>24.998</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-worker-3.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>24.999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-worker-4.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>24.999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-worker-5.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>24.999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-worker-6.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>24.999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-worker-7.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-worker-8.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>worker</td>
<td>24.999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gtp-viya-controller.gtp-mericas.sashq-d.openstack.sas.com</td>
<td>controller</td>
<td>25.102</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**Best Practices for Migrating SAS Code to Leverage CAS**

**Appendix – How to Load a SAS7BDAT into a CAS Table**

```sas
proc cas;
  table.dropCaslib /
    caslib='sas7bdat' quiet = true;
  table.addCaslib /
    caslib="sas7bdat"
    dataSource={srctype="path"}
    path="/SAS Viyafiles/sasss1/data";
run;

/*/ **SAS7BDAT datasets will be loaded in parallel** when the shared file system has been mounted on all CAS workers and all CAS workers have direct access to the SAS7BDAT file at the physical path /SAS Viyafiles/sasss1/data */

proc casutil;
  load casdata="dummy.sas7bdat" outcaslib="casuser“ casout="dummy“ replace
    importoptions=(filetype="basesas" dtm="auto“ debug="dmsglvli");
run;
```
libname CASWORK cas caslib=casuser;

options USER = CASWORK;

proc cas;
   table.dropCaslib / caslib='sas7bdat' quiet = true;
   table.addCaslib / caslib="sas7bdat"
   dataSource={srctype="path"}
   path="/SAS Viyafiles/sasss1/data";
run;
quit;

proc casutil;
   save casdata="baseball" incaslib="casuser"
   casout="baseball.sas7bdat" outcaslib="sas7bdat"
   exportoptions={filetype="BASESAS"}
   replace;
quit;
Appendix – How to Load a SASHDAT into a CAS Table

```sas
proc cas;
  table.dropCaslib / caslib="sashdat" quiet = true;
  table.addCaslib / caslib="sashdat"
    dataSource={srctype="DNFS"}
    path="/SAS Viyafiles/sasss1/data";
quit;

proc casutil;
  load casdata="baseball.sashdat" outcaslib="casuser" casout="baseball" replace;
quit;
```

/* DNFS is intended for parallel-only transfer. Therefore, the SAS log does not indicate the actual dataTransferMode used to load the data into CAS. The definition of DNFS means that data transfers in parallel such that the CAS workers participate in working with the data directly from source. When using the DNFS type of CASLIB, the assumption is that data movement will be scalable and performant due to the underlying shared file system technology used */
How to Save a CAS Table to SASHDAT

```
libname CASWORK cas caslib=casuser;
options USER = CASWORK;
caslib _all_ assign;
data casuser.baseball;
    set sashelp.baseball;
run;
proc cas;
    table.dropCaslib / caslib="sashdat" quiet = true;
    table.addCaslib / caslib="sashdat"
dataSource={srctype="DNFS"}
    path="/SAS Viyafiles/sass1/data";
run;
quit;

proc cas;
    table.save / caslib="sas7bdat"
    table={name="baseball", caslib="casuser"},
    name="baseball.sashdat"
    replace=True;
quit;

proc casutil;
    save casdata="baseball" incaslib="casuser"
    casout="baseball.sashdat" outcaslib="sashdat"
    replace;
quit;
```
Best Practices for Migrating SAS Code to Leverage CAS
Appendix – How to Change the Active CASLIB

/* Set active CASLIB to CASUSER */
options caslib="casuser";

OR

proc cas;
   sessionProp.setSessOpt /
      caslib="casuser";
   run;
quit;
Best Practices for Migrating SAS Code to Leverage CAS
Appendix – How to Leverage One Level CAS Table Names

/* LIBNAME using the CAS engine */
libname CASWORK cas caslib=casuser;
/* Changing the default location of all one level named tables */
/* from SASWORK to CASWORK */
options USER = CASWORK;
/*Binds all CAS librefs and default CASLIBs to your SAS client */
caslib _all_ assign;
proc cas;
  table.dropCaslib/
    caslib='sas7bdat' quiet = true;
  table.addCaslib/
    caslib="sas7bdat" dataSource={srctype="path"}
    path="/SAS Viyafiles/sasss1/data";
run;

proc casutil;
  load casdata="baseball.sas7bdat"
    casout="baseball" outcaslib="casuser" replace
    importoptions=(filetype="basesas" dtm = "auto");
rung;
/*Runs in CAS*/
/*Target table is a CAS Table*/
/*Source table is a CAS Table*/
data baseball2;
  set baseball;
  ratio = nruns / nhits;
rung;
/*Runs in CAS*/
/*Source table is a CAS Table*/
/*PROC MEANS is CAS enabled*/
proc means data=baseball2;
rung;
libname CASWORK cas caslib=casuser;
options USER = CASWORK;
caslib _all_ assign;
/* Load SAS7BDAT into CAS */
proc cas;
  table.dropCaslib /
    caslib='sas7bdat' quiet = true;
  table.addCaslib /
    caslib="sas7bdat" dataSource={srctype="path"}
    path="/SAS Viyafiles/sasss1/data";
run;
proc casutil;
  load casdata="cars.sas7bdat"
  casout="cars" outcaslib="casuser" replace
    importoptions=(filetype="basesas" dtm= "auto");
run;

/* Create a CAS view */
/* For each DESCENDING numeric create a new variable(s) */
/* The value of the new variable(s) is the negated value */
/* of the original DESCENDING BY numeric variable(s) */
proc cas;
  table.view / replace = true
    caslib='casuser'
    name='descending'
    tables={{
      name='cars'
      varlist={"msrp" "make"},
      computedVars={{name='n_msrp'},
        computedVarsProgram='n_msrp = -(msrp)'}
    }};
run;
quit;
Best Practices for Migrating SAS Code to Leverage CAS

Appendix – How to Emulate DATA Step with DESCENDING BY for Numeric Variables

/* CAS View is the source table */
/* Use the negated variables in the view on the BY (ascending) */
/* Now the original variable(s) are in descending order */

data descending2;
  set descending;
  by make n_msrp;
  if first.make;
run;
libname caswork cas caslib=casuser;
options user=caswork;

**data** caswork.baseball(partition=(\texttt{div row\_order}) orderby=(\texttt{div row\_order}));
  set sashelp.baseball;
  row_order = _n_; 
run;

**data** caswork.baseball2;
  set caswork.baseball;
  **BY** DIV ROW\_ORDER; /* The data is already in the correct order */
run;
libname caswork cas caslib=casuser;
options user=caswork;

data caswork.baseball(partition=(div row_order) orderby=(div row_order));
  set sashelp.baseball;
  row_order = _n_; 
run;

data caswork.baseball2;
  set caswork.baseball;
  BY DIV TEAM ROW_ORDER; /* The BY is done on the fly and consumes resources */
run;
/*LIBNAME using the CAS engine*/
libname CASWORK cas caslib=casuser;

/*Changing the default location of all one level named tables*/
/*from SASWORK to CASWORK*/
options USER = CASWORK;

caslib _all_ assign;

/* Load SAS7BDAT into CAS */
/* Requires CASLIB with a path to baseball.sas7bdat */
proc cas;
    table.loadTable /
        path="baseball.sas7bdat"
        casout={caslib="casuser",
            name="baseball", replace=true};
quit;

ods csvall file="/SAS Viyafiles/sasss1/ODS_CSVALL.csv";
proc print data=caswork.baseball;
run;
ods csvall close;
Best Practices for Migrating SAS Code to Leverage CAS
Appendix – How to use FEDSQL with Descending Order BY

```sas
/* cas casauto terminate; */
cas;
caslib _all_ assign;

/* Load SAS7BDAT into CAS */
proc cas;
  table.loadTable /
    path=“baseball.sas7bdat"
  casout={caslib="casuser", name="baseball", replace=true};
quit;

/* Load SAS7BDAT into CAS */
proc cas;
  table.loadTable /
    path=“baseball.sas7bdat"
  casout={caslib="casuser", name="baseball", replace=true};
quit;

proc fedsql sessref=casauto;
  create table baseball_desc as
    select distinct a.*
    from baseball a
    order by team DESC;
quit;
```
/*proc sort data=work.baseball
   out=work.nodup nodupkey;
   by DIV TEAM;
run;*/

data caswork.nodup;
   set caswork.baseball;
   by div team;
   if first.team then output;
run;

proc fedsql sessref=mysess;
   create table nodup {options replace=true}
      as select distinct *
      from baseball
      group by div team;
quit;
PROC SORT DATA=CASUSER.table OUT=LIBREF.table;
   BY highCardinality;
run;

DATA LIBREF.table2;
   SET LIBREF.TABLE;
   BY highCardinality;
run;

PROC CASUTIL;
   load casdata="table2.sas7bdat"
      outcaslib="casuser" casout="table2"
      replace
      importoptions=(filetype="basesas "
                     dtm= "auto");
run;
/* Load SAS7BDAT into CAS */
/* Requires CASLIB with a path to cars.sas7bdat */

proc cas;
    table.loadTable /
        path="cars.sas7bdat"
    casout={caslib="casuser",
        name="cars", replace=true};
quit;
proc format library=work.formats caslib="casformats"
    value enginesize
        low - <2.7 = "Very economical"
        2.7 - <4.1 = "Small"
        4.1 - <5.5 = "Medium"
        5.5 - <6.9 = "Large"
        6.9 - high = "Very large";
run;

/* Save format so they can be used between CAS sessions */
cas casauto savfmtlib fmtlibname=casformats
    table=enginefmt replace;

proc print data=caswork.cars;
    format enginesize enginesize.;
    var enginesize;
run;
/*LIBNAME using the CAS engine*/
libname CASWORK cas
  caslib=casuser;

/*Changing the default location of all
  one level named tables from SASWORK
  to CASWORK*/
options USER = CASWORK;

/*Binds all CAS librefs and default
CASLIBs to your SAS client*/
caslib _all_ assign;

/*Runs in CAS*/
data baseball (append=yes);
  set baseball2 baseball3;
run;
90 data baseball (append=yes);
91 set baseball2 baseball3;
92 run;

NOTE: Running DATA step in Cloud Analytic Services.
NOTE: The DATA step will run in multiple threads.
NOTE: There were 322 observations read from the table BASEBALL2 in caslib CASUSER(sasss1).
NOTE: There were 322 observations read from the table BASEBALL3 in caslib CASUSER(sasss1).
NOTE: The table baseball in caslib CASUSER(sasss1) has 966 observations and 24 variables.
NOTE: The APPEND operation for table baseball in caslib CASUSER(sasss1) added 644 observations.
NOTE: DATA statement used (Total process time):
real time 1.11 seconds
cpu time 0.01 seconds
libname CASWORK cas caslib=casuser;
options USER = CASWORK;
caslib _all_ assign;

PROC DELETE data=CASWORK.TABLE1 CASWORK.TABLE2;
run;

proc casutil
   sessref=mysess
   incaslib="caswork";
   droptable casdata="table1";
   droptable casdata="table2";
run;

proc cas;
   sessref=mysess;
   table.droptable / caslib="caswork" name="table1";
   table.droptable / caslib="caswork" name="table2";
run;
Reading Materials
Best Practices for Migrating SAS Code to Leverage CAS

Reading Material

1. Parallel Programming with the DATA Step: Next Steps (2018) required reading
2. How to achieve repeatable results with distributed DATA Step BY Groups (2018) required reading
3. The following Base SAS procedures are CAS Enabled (2018)
11. Six reasons you should stop using the RANUNI function to generate random numbers (2013) required reading
Next Steps
Best Practices for Migrating SAS Code to Leverage CAS

Next Steps

• How do we get started transitioning our code from SAS9 to SAS Viya?
  • Who can help us?
• We store data in multiple locations.
  • How do we get this data into CAS?
• Why are we converting PROC SQL to FEDSQL?
• How do I know what parts of the DATA Step can run in CAS and what parts will send the data back to the v9 engine for processing?
• If my SAS9 PROC isn’t CAS-compliant and there’s no CAS equivalent, what are my options?
• If I have to use a CAS-equivalent PROC instead of my SAS9 PROC, who can help me with this?
• How do I obtain the SAS Viya Readiness Assessment Utility?

Contact your SAS account executive and they will contact Brian and Steven to support your migration of SAS code to leverage CAS