Paper SAS158-2022

Cooking In SAS® Viya

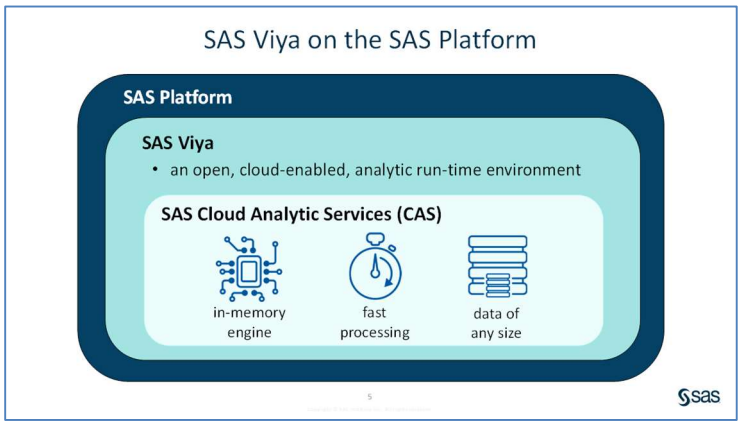
Charu Shankar, SAS Institute Inc.

ABSTRACT

Can I use my existing SAS code in SAS Viya? What's a Cloud Analytic Server(CAS)? How's SAS 9 different from CAS. What is a Compute Server? What is the advantage of learning to speak in SAS Viya? These are some of the questions top of mind for the Classic SAS 9 User who has a solid foundation in SAS 9 & isn't sure how things integrate with existing SAS code. Learn through the yummy analogies of SAS instructor-yoga teacher & chef, Charu who drew from her experience of cooking in a yoga retreat in the Bahamas for over 300 guests to understand the distinction between SAS & CAS. You too can learn the concepts. if techie terminology has your head all wound up in circles, then this session is just for you. Learn how you can stretch your knowledge of SAS programming concepts to beautifully write CAS code with just a few simple tweaks. All levels welcome. Some basic knowledge of SAS programming will help you get more value out of this session.

INTRODUCTION

SAS Viya is a cloud-enabled, in-memory analytics engine that provides quick, accurate and reliable analytical insights. The latest enhancement of the SAS platform, SAS Viya is an open, cloud-enabled, analytic runtime environment with a number of supporting services. One of those supporting services is SAS Cloud Analytic Services, or CAS. CAS provides a powerful in-memory engine that delivers blazing speed to accurately process your big data. It uses scalable, high-performance, multi-threaded algorithms to rapidly perform analytical processing on in-memory data of any size. This paper will walk you through the fundamentals of understanding CAS in relation to SAS.



TERMINOLOGY

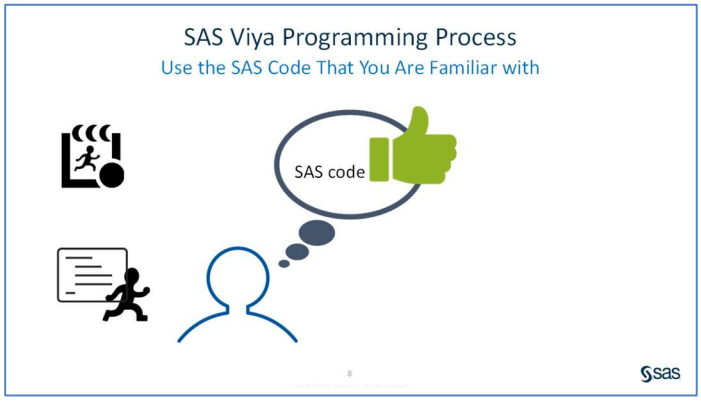
File is used to refer to the source data that is in a caslib's data source. For a caslib that uses a path-based data source, this is natural. For a caslib that uses a database as a data source, the tables in the database are referred to as files.

Table is used to refer to in-memory data. After a file (using the preceding definition) is loaded into the server, it is referred to as a table.

CAS - in Viya, SAS Cloud Analytic Services (CAS) is the star of the show, providing lightning, fast analytics of in-memory data for SAS Visual Analytics and other software offerings.

SPRE - Foundation SAS, the long-time workhorse of SAS analytics is also offered, referred to as the SAS

Programming Runtime Environment (SPRE). SPRE provides a user interface and data processing environment for executing classic SAS program code. It offers the Foundation SAS software we're all familiar with, including Base SAS, SAS/ACCESS engines, , and more as well as the SAS Studio web application.



1. CONNECT TO CAS

Once a CAS session starts, you can write code and submit CAS enabled procedures.

CONNECT TO CAS SERVER

Code to connect to CAS Server:

```
cas mySession sessopts=(caslib=casuser timeout=1800);
```

```
NOTE: 'CASUSER(student)' is now the active caslib.
NOTE: The CAS statement request to update one or more session options for session MYSESSION
completed.
```

Display 1: Log showing that CAS Session MYSESSION successfully started

ACCESSING CASLIBS

A Caslib is a container for both the files in the Caslib's data source and the in-memory tables that you load from the data source.

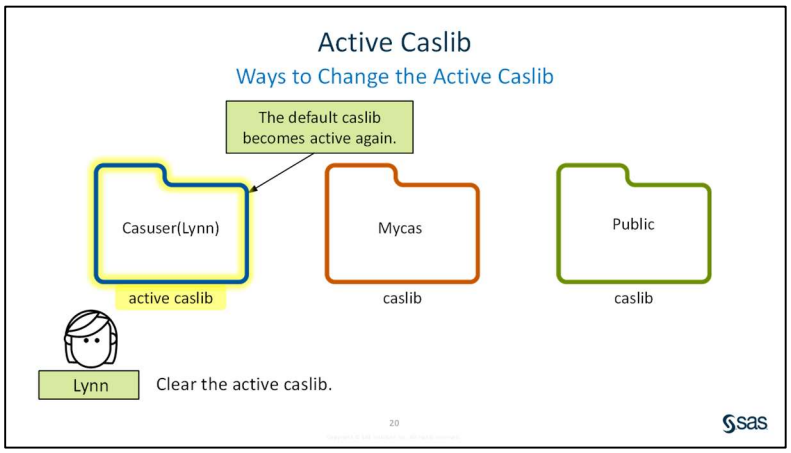
Code to list CAS libraries:

```
Caslib _all_ list;
```

```
78   caslib _all_ list;
NOTE: Session = MYSESSION Name = CASUSER(student)
         Type = PATH
         Description = Personal File System Caslib
         Path = /cas/data/caslibs/casuserlibraries/student/
         Definition =
         Subdirs = Yes
         Local = No
         Active = Yes
         Personal = Yes
NOTE: Action caslib LIST completed for session MYSESSION.
79
80   %studio_hide_wrapper;
```

Display 2: Log verifying that CASUSER is the active Caslib.

CHANGING ACTIVE CASLIB



There is only one active caslib at a time in a CAS session. The active caslib is where data is processed by default.

Code to list the files in Casuser, default Caslib or active Caslib:

```
proc casutil;
   list files;
quit;
```

The CASUTIL Procedure

CAS File Information

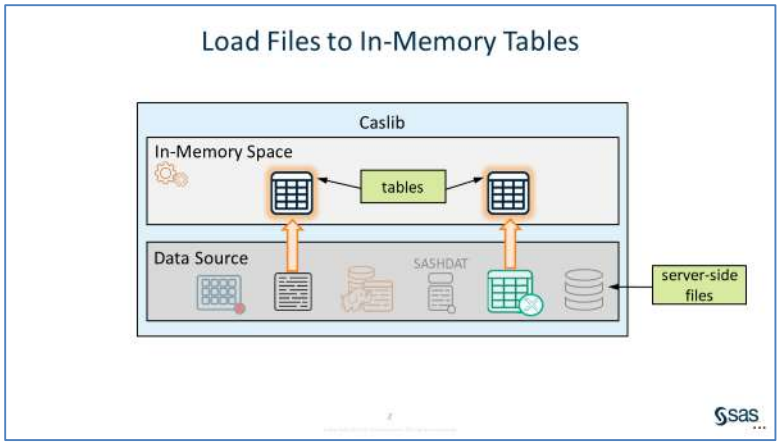
Name	Permission	Owner	Group	Encryption Method	File Size	Last Modified (UTC)
orders_hd.sashdat	-rwxr-xr-x	cas	sas	NONE	1.6GB	06SEP2022:18:10:
products.xlsx	-rwxr-xr-x	cas	sas		219.8KB	06SEP2022:18:11:
sales.csv	-rwxr-xr-x	cas	sas		10.3KB	06SEP2022:18:11:
sales.sas7bdat	-rwxr-xr-x	cas	sas		72.0KB	06SEP2022:18:11:

Display 3: Results Tab Displaying List Of Files in Casuser.

2 LOAD DATA TO A CASLIB AND PROCESS DATA IN CAS

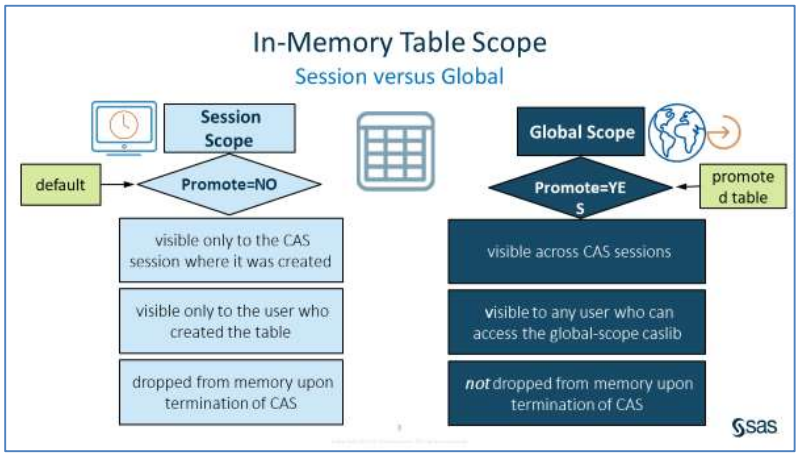
In the first section, we started a CAS session and accessed a CAS library. Since CAS can only process data in its in-memory space, the next step is to load your file into memory. Subsequently, data updates and analysis can begin.

Source data files mapped to a Caslib are referred to as “server-side” files. These files can be rapidly loaded into the Caslib’s in-memory space for processing. Once a file is loaded into memory it is referred to as a table. These CAS tables are in-memory *copies* of the associated CAS file - the source data files remain on disk and unchanged.



SESSION SCOPE VS. GLOBAL SCOPE

In memory tables can have either session or global scope.



By default, In-memory tables have session scope. A session-scope table is only accessible in the CAS session where it was created. It's only visible to the user who created it. Session-scope tables are useful for ad hoc data access and analysis because they don't require access control checks or locking for concurrent access.

A session-scope in-memory table only exists for the duration of the session. When the CAS session ends, the table is dropped.

To share data across your sessions, or with other users, create a global-scope table, also called a promoted table. You can promote a table when you load a file into memory or promote an in-memory session table. After a session-scope table is promoted, it's visible across CAS sessions.

Unlike session-scope tables, global-scope tables are not dropped from memory when a CAS session ends. The table is still available to other sessions and will be available in the next CAS session the user starts.

To understand the concepts of session vs. global scope, we will load the client-side SAS data set, **mysas.employees**, into the personal Caslib. We creates an in-memory session-scope table, **myemployees.** This is a table that others will need to access. Then Use the PROMOTE statement to

create a global-scope table in the **Public** caslib so that other users can use the in-memory table.

Code to load data, create a session scope table and promote it

```
proc casutil;
   load data=sashelp.cars outcaslib=casuser
        casout="MyCars" replace;
   load data=pvbase.employees outcaslib="casuser"
          promote ;
quit;
proc casutil;
   list tables incaslib="casuser";
quit;
```

The log shows that CAS processed the code, and the results show specific metadata about the **myemployees and mycars** tables.

The CASUTIL Procedure

Table Information for Caslib CASUSER(student)

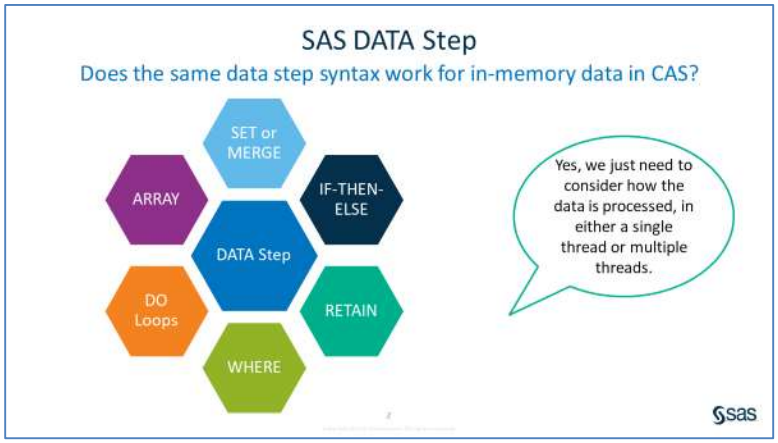
Table Name	Label	Number of Rows	Number of Columns	Indexed Columns	NLS encoding	Created	Last Modifi
MYCARS	2004 Car Data	428	15	0	utf-8	2022-09-07T18:01:51+00:00	2022-09-07
EMPLOYEES		1048	10	0	utf-8	2022-09-07T18:01:51+00:00	2022-09-07

Display 4: Metadata about the myemployees and mycars tables

The **mycars** table is session scope (**Promoted Table=No**) and will be dropped from memory when the CAS session ends.

The **employees** table is global scope (**Promoted Table=YES**) and will not be dropped from memory when the CAS session ends.

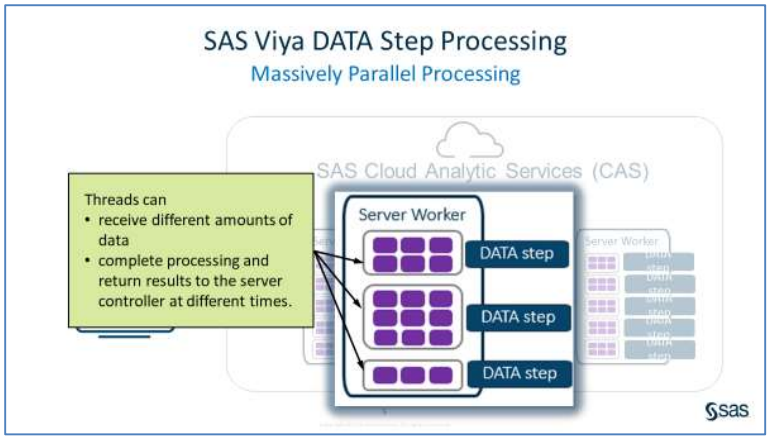
3 MODIFYING SAS® PROGRAMS TO RUN IN SAS® VIYA®



When a DATA step is executed in Base SAS, it runs in a single thread on the SAS workspace server. Processing data in a single thread reads data sequentially, one row at a time.

SAS Viya enables data to be divided and processed simultaneously on multiple threads. When a data step executes in CAS, each thread executes the program statements on its data, and returns the results to the controller.

The threads might receive different amounts of data, and might complete their processing and return the results in a seemingly random order. SAS Viya reassembles the results. We'll look at examples where the parallel processing is transparent to the user. The only difference you'll see is faster execution. We'll also look at situations where you, as the programmer, will need to take additional action to summarize the results from the threads.



One big difference is the fact that **_THREADID_** is equal to different values for each row in the log. The threads operate independently. Therefore, the log messages were generated by each thread at slightly different times. The values represent the thread the DATA step was executed on in the CAS session. There are 16 threads available (_NTHREADS_=16). In this execution of the code, thread 3 completed the execution first, and then thread 8, and so on.

If you run the program multiple times, you might get a different order each time the program run. This is exactly what we want to happen when a program is executed in multiple threads. Otherwise, the performance gains by threading are lost if the DATA step were to somehow synchronize the output to the log.

```
NOTE: Running DATA step in Cloud
Analytic Services. Processed on
_THREADID_=3 _NTHREADS_=16

Processed on _THREADID_=8
_NTHREADS_=16 Processed on
_THREADID_=5 _NTHREADS_=16
Processed on _THREADID_=7
_NTHREADS_=16 Processed on
_THREADID_=4 _NTHREADS_=16
Processed on _THREADID_=9
_NTHREADS_=16 Processed on
_THREADID_=1 _NTHREADS_=16
Processed on _THREADID_=15
_NTHREADS_=16 Processed on
 THREADID =11
```

Display 5: Log indicating multi-thread processing in the CAS Session

MODIFYING DATA STEP CODE TO RUN IN VIYA – NEW VARIABLES

Sometimes to get the DATA step to process in CAS, it's as simple as modifying the library reference on the DATA statement and the SET statement to use a caslib. When both the output and input tables are CAS tables, the DATA step will process in CAS.

Let's look at a DATA step. We will modify the Base SAS DATA step to run in multi-threaded environment in CAS.

```
78   data work.shipping;
79       set pvbase.orders end=eof;
80       where OrderType ne 1;
81       DaysToDeliver=Delivery_Date-Order_Date;
82       drop xy: Discount Employee_ID;
83       if eof then put _threadid_=   _N_=;
84   run;
_THREADID_=1 _N_=235699
NOTE: There were 235699 observations read from the data set PVBASE.ORDERS.
      WHERE OrderType not = 1;
NOTE: The data set WORK.SHIPPING has 235699 observations and 21 variables.
NOTE: DATA statement used (Total process time):
      real time           0.79 seconds
      cpu time            0.79 seconds
```

Display 6: Log showing program runs in a single thread.

Code to modify data step to run in CAS.

```
84   data casuser.shipping;
85       set casuser.orders end=eof;
86       where OrderType ne 1;
87       DaysToDeliver=Delivery_Date-Order_Date;
88       drop xy: Discount Employee_ID;
89       if eof then put _threadid_=   _N_=;
90   run;
NOTE: Running DATA step in Cloud Analytic Services.
NOTE: The DATA step will run in multiple threads.
_THREADID_=9 _N_=13107
_THREADID_=11 _N_=16696
_THREADID_=6 _N_=15041
_THREADID_=15 _N_=13166
_THREADID_=2 _N_=16256
_THREADID_=14 _N_=15379
_THREADID_=13 _N_=14665
_THREADID_=10 _N_=14008
_THREADID_=8 _N_=16152
_THREADID_=3 _N_=15049
_THREADID_=1 _N_=15775
_THREADID_=4 _N_=14837
_THREADID_=12 _N_=14190
_THREADID_=7 _N_=14208
_THREADID_=16 _N_=12073
_THREADID_=5 _N_=15097
NOTE: There were 235699 observations read from the table ORDERS in caslib CASUSER(student).
NOTE: The table shipping in caslib CASUSER(student) has 235699 observations and 21 variables.
NOTE: DATA statement used (Total process time):
      real time           0.40 seconds
```

Display 7: Log showing program runs in multiple threads.

The data was distributed across the 16 threads in the CAS session. The results were returned as each thread completed its processing. Thread 9 completed first after processing 13107 rows, and then thread 11, and so on. If you were to add up all the values of **_N_**, the sum would equal 235699, which is the total number of rows that were read from the **casuser.orders** table.

SUMMARIZING IN CAS USING THE SUM STATEMENT

Code to show data step running in compute server.

```
/* DATA step in Compute Server */
data work.eurorders;
   set pvbase.orders end=eof;
      if Continent="Europe" then EurOrders+1;
         if eof=1 then output;
            keep EurOrders;
run;
```

Display 8: Log showing program runs in multiple threads.

Let's start by looking at the first DATA step in this program. The IF-THEN statement creates an accumulating column that counts the number of rows in which **Continent** equals *Europe*. It then outputs the last row to get the total number of orders from Europe. It will run on the SAS Compute Server and process the data in a single thread.

The output table shows that there were 653,684 orders from Europe, and the log shows that the step took .5 seconds to run.

Note: Your time values might differ.

To generate the same results in CAS, WE need to use two steps.

Code to show CAS step.

```
 /* Two DATA steps in CAS */
 /* DATA step 1 */
   data casuser.eurorders_thread;
      set casuser.orders end=eof;
         if Continent="Europe" then EurOrders_thread+1;
            if eof=1 then output;
               keep EurOrders_thread;
   run;

 /* DATA Step 2 */
   data casuser.eurorders / single=yes;
      set casuser.eurorders_thread end=eof;
         EurOrders+EurOrders_thread;
            keep EurOrders;
               if eof then output;
   run;
```

The first DATA step is similar to the previous step, but it references in-memory tables in both the DATA and SET statements. By default, it processes in multiple threads, so the output data will include multiple rows, one from each thread. The second DATA step sums the multiple rows from **casuser.eurorders** into one row using a single thread.

I'll highlight and run the two DATA steps that will run in CAS. The first output table has one row per thread, and the output table has the total number of orders.

In the log, note the values for real time and CPU time. Although these steps are taking only a few seconds to run, the combined real time for the two DATA steps was a fraction of the processing time for the single thread. The time savings would likely be more dramatic when processing extremely large

tables or complex code.

Although processing times can vary from run to run, the relationship between the runs should remain the same. Therefore, we can conclude that, in general, the two-DATA-step process is faster than using one single-threaded DATA step.

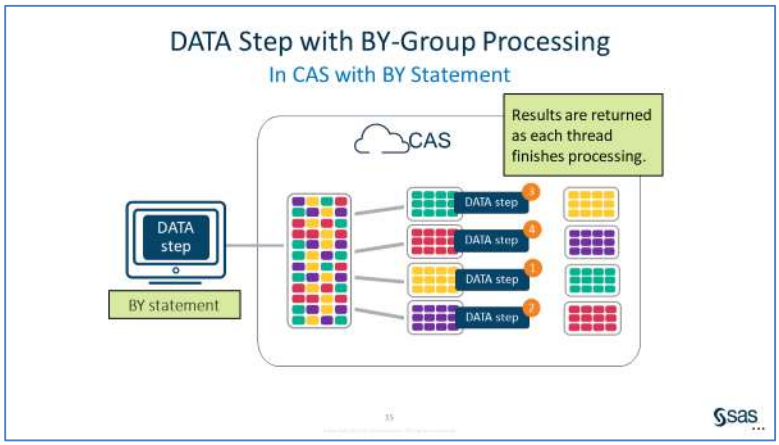
MODIFYING DATA STEP CODE TO RUN IN VIYA – BY STATEMENT

If using the DATA step to process in groups or merge data based on the value of variable(s), then you would have to first sort the data and then use the BY statement and FIRST. and LAST. processing to identify the first and last row in each group. Sorting can be a very resource intensive, especially with very large data sets and when the DATA step is processed in Base SAS, the rows are processed sequentially in a single thread.

In Cas, the default when data is loaded into CAS is to distribute the input data based on the original order among the different threads or multiple machines. The data step is executed among the different threads or on multiple machines.

When a BY statement is added to the DATA step, the rows are group based on the first BY variable and then distributed across multiple threads or machines. And because the data is distributed based on the value of the BY variable, PROC SORT is no longer necessary.

The DATA step with the BY statement executes on each thread. Results are returned as each thread finishes processing. Thread three processes the data step and returns the results first. The order might be different each time the program executes.



CONCLUSION

This paper attempted to showcase the power of Viya and CAS from assigning libraries, to moving data and manipulating it. Performance benefits were highlighted so that the reader weighing options can perhaps begin to consider Viya for their daily data work.

ACKNOWLEDGEMENTS

The author is grateful to the many SAS users that have entered her life. Charu is grateful to the Western Users of SAS Software Committee for the opportunity to present this paper. She would also like to express her gratitude to her manager, Stephen Keelan without whose support and permission, this paper

would not be possible.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Charu Shankar
SAS Institute Canada, Inc.
Charu.shankar@sas.com
https://blogs.sas.com/content/author/charushankar/

REFERENCES

CAS Concepts
https://go.documentation.sas.com/?docsetId=calserverscas&docsetTarget=n05000viyaservers000000admin.htm&docsetVersion=3.4&locale=en
An intro to Viya Programming
https://go.documentation.sas.com/api/docsets/pgmdiff/3.4/content/pgmdiff.pdf
Differences in the SAS® 9 and SAS® Viya™ 3.1 Platforms
https://go.documentation.sas.com/api/docsets/whatsdiff/3.1/content/whatsdiff.pdf?locale=en#nameddest=n0evbd1ha0clqvn1sbz5yag06xi6
SPRE (SAS Programing Runtime Environment)
https://communities.sas.com/t5/SAS-Communities-Library/Deploying-the-SPRE-in-SAS-Viya-3-4/ta-p/602891
SAS® Cloud Analytic Services 3.1: Language Reference
https://go.documentation.sas.com/api/docsets/casref/3.1/content/casref.pdf?locale=en#nameddest=p05ccny5glgvwan19mkisxi8z1jk