Ask the Expert Special Edition

Programming in SAS® Viya®

Peter.Styliadis@sas.com

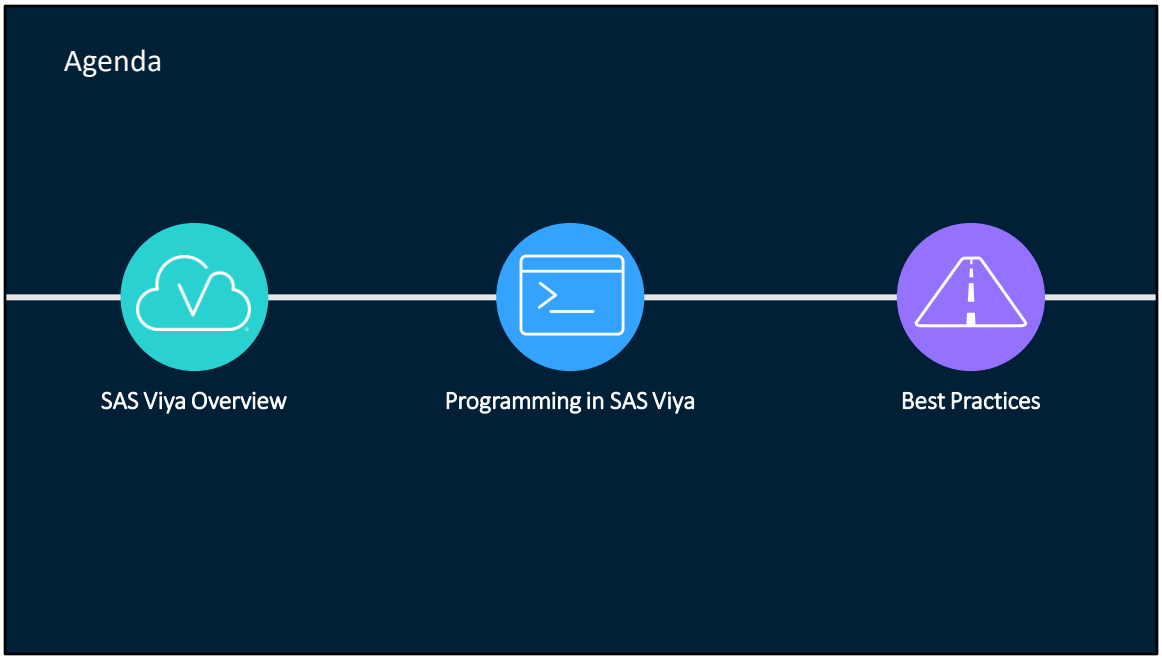
Sr Technical Training Consultant

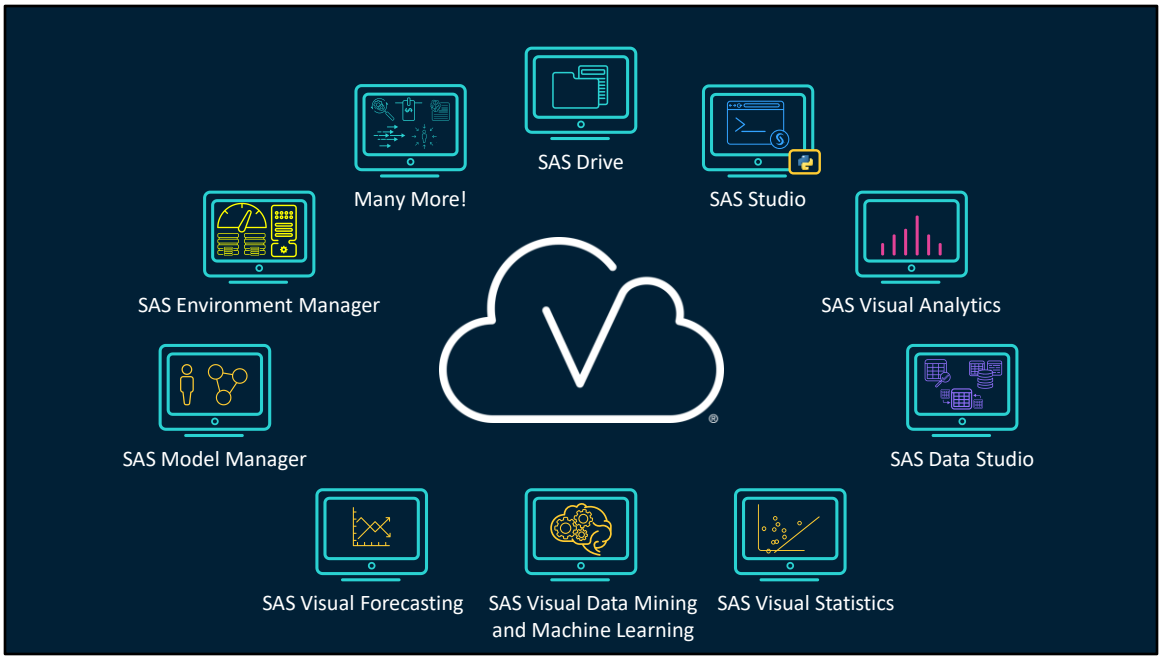
Connect with me on LinkedIn!

Target Audience

SAS Programmer

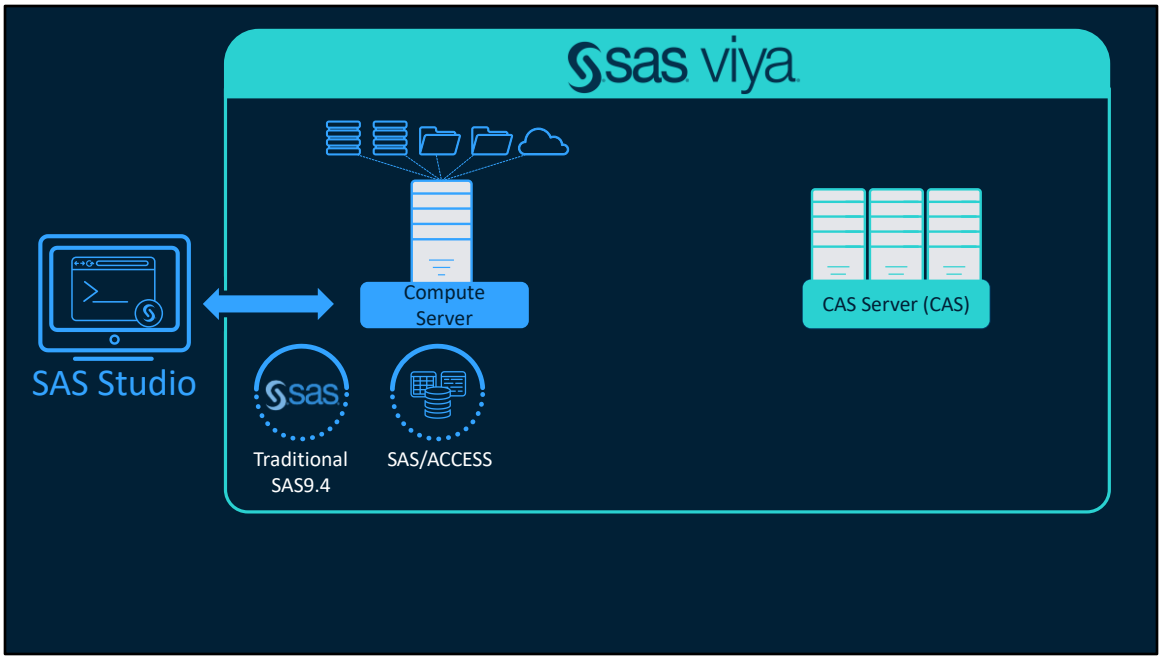
This presentation is focused on SAS programmers who are transitioning to SAS Viya. While experienced SAS programmers are they target audience, all can attend. Basic SAS programming knowledge is assumed throughout this presentation.


Agenda
SAS Viya Overview
Programming in SAS Viya
Best Practices



SAS Viya consists of many applications that enable you to work with your data no matter your job role or skill level. This can be done through the collection of integrated AI, analytic, and data management solutions in SAS Viya.

This presentation will focus on using SAS Studio to execute programs in SAS Viya.



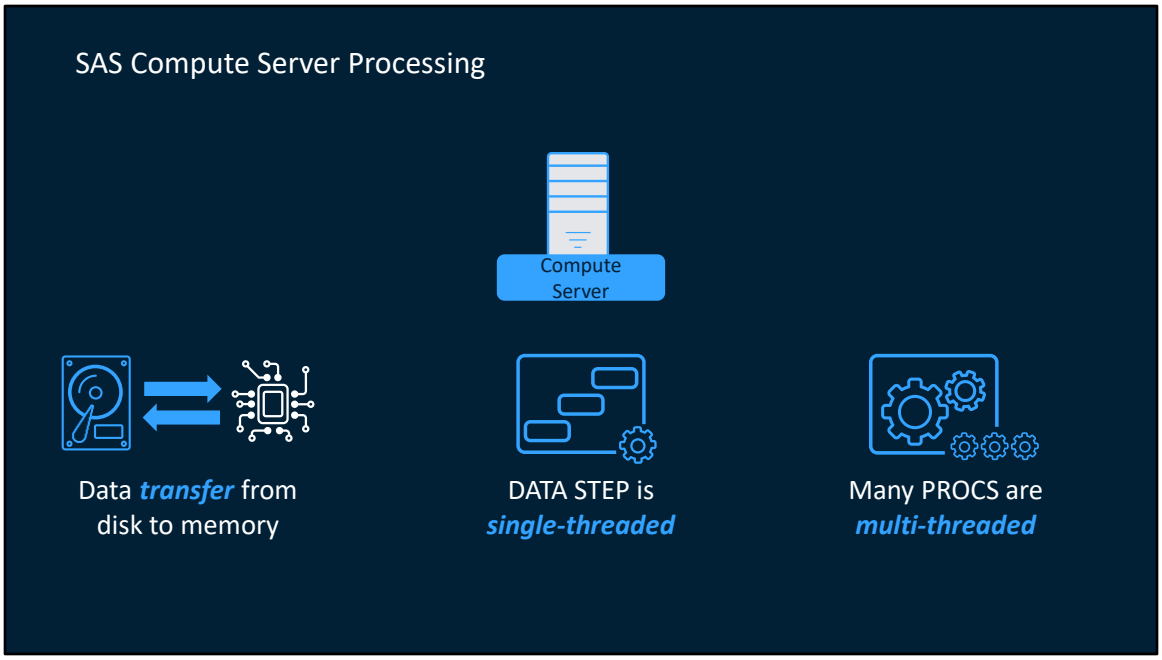
Before we dive into using programming using SAS Studio, I need to give you a quick, high-level introduction into the SAS Viya architecture.

First, SAS Viya contains two servers for working with your data.

- The first is called the SAS Compute Server (which is like the SAS9 Workspace server for those of you familiar with SAS9)
- The second is Cloud Analytics Services, or the CAS server for short. A massively parallel processing environment (MPP) in SAS Viya.

To access SAS Viya, you log in using your browser and select an application. To access SAS Studio, you select Develop SAS Code. SAS Studio accesses SAS Viya through the compute server.

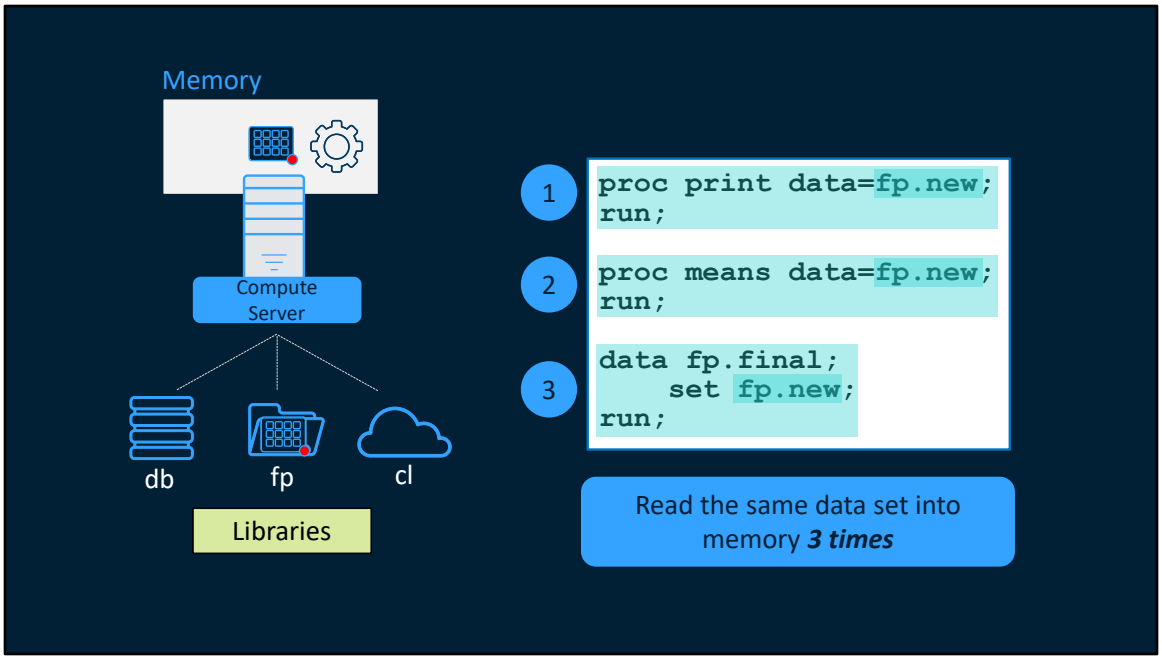
- The compute server works like the traditional SAS9 workspace server. It is used to execute traditional SAS code that you've always used. Do you have programs that you want to execute as they have been for years? Use the SAS compute server.
- The SAS compute server accesses data throughout your organization using the traditional SAS/ACCESS engine.



To understand the advantages and limitations of CAS architecture, it's important to first understand the SAS compute server processing. For some of you, this might be review.

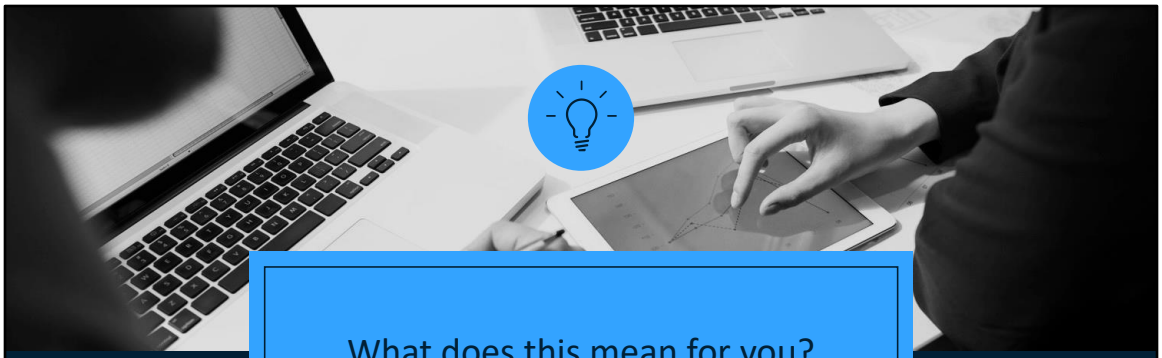
1. Typically, SAS was installed on a system with dedicated disk, RAM and CPU. When requested by a SAS program, the operating system reads data from the disk in blocks into memory. When the processing is complete, memory is cleared by SAS.
2. When using the DATA step, SAS gets the data from memory in a single read-write thread and passes the data to the DATA step to process one row at a time in a single compute thread, which leverages only one of the CPUs available. When finished, SAS passes the data back to the operating system buffer via the single read-write thread, and the system writes the data back to disk in blocks.
3. Certain Base SAS procedures have algorithms that can take advantage of threaded processing. These procedures are thread-enabled to split parts of the procedure algorithm so that it executes some parts of the algorithm in threads. For example, the SORT procedure is thread-enabled so that the sorting takes place in available threads and each thread sorts a part of the data. The procedure then quickly generates the data set in sorted order from the multiple threads. The following Base SAS procedures support threaded processing:
    - MEANS
    - REPORT
    - SORT
    - SUMMARY
    - TABULATE
    - SQL

SAS Documentation:
https://go.documentation.sas.com/doc/en/pgmsascdc/v_027/proc/n0n5mm9l2pmpevn1lsjqccmgp8tx.htm#p0fbq4inzwf57bn1xrcss87xxj8t



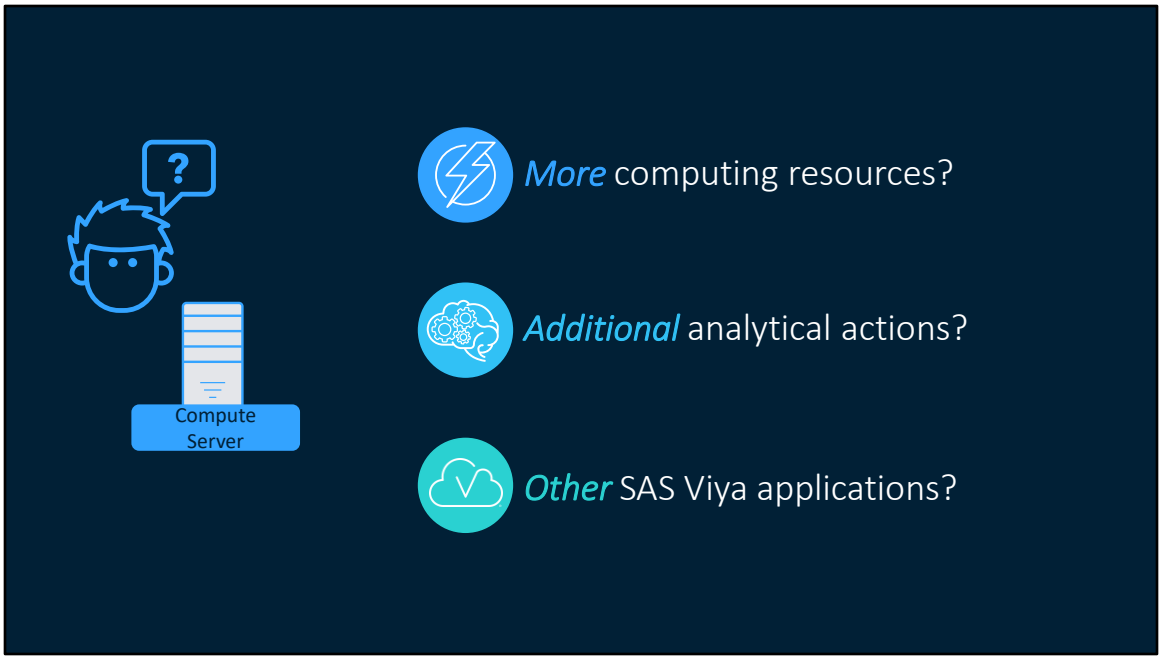
When executing a SAS program in the compute server, each step will read the data into memory for you. You don't have to think about how to load and unload it from memory. This is massive benefit because it allows you to not worry about loading data into memory or using up all your memory when processing data since it's automatically cleared for you.

However, this also means that you have limited control on how the data gets read into or cleared from memory. This can cause multiple reads of the same table. Otherwise know as or input and output (I/O).



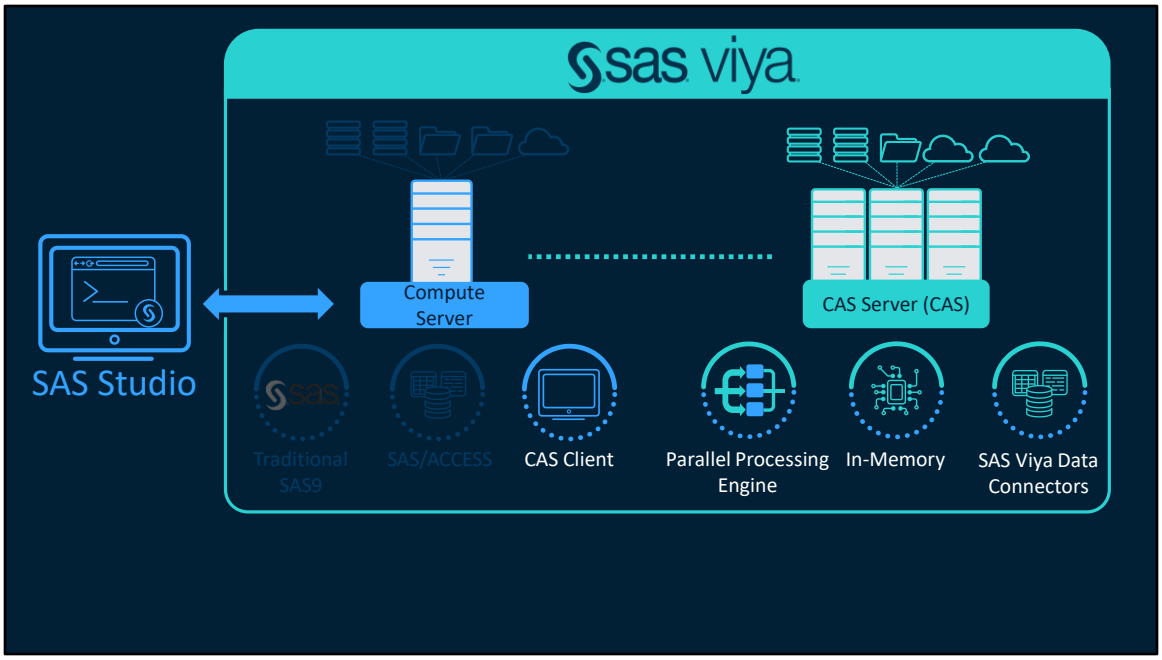
What does this mean for you?

Use your ***familiar SAS*** knowledge!



While the SAS compute can process data effectively, what if

- your program takes a long time to execute, and you need more computing resources?
- you need additional analytical actions for reading in data, machine learning, deep learning and image analysis?
- you want to use other SAS Viya applications like Visual Analytics?

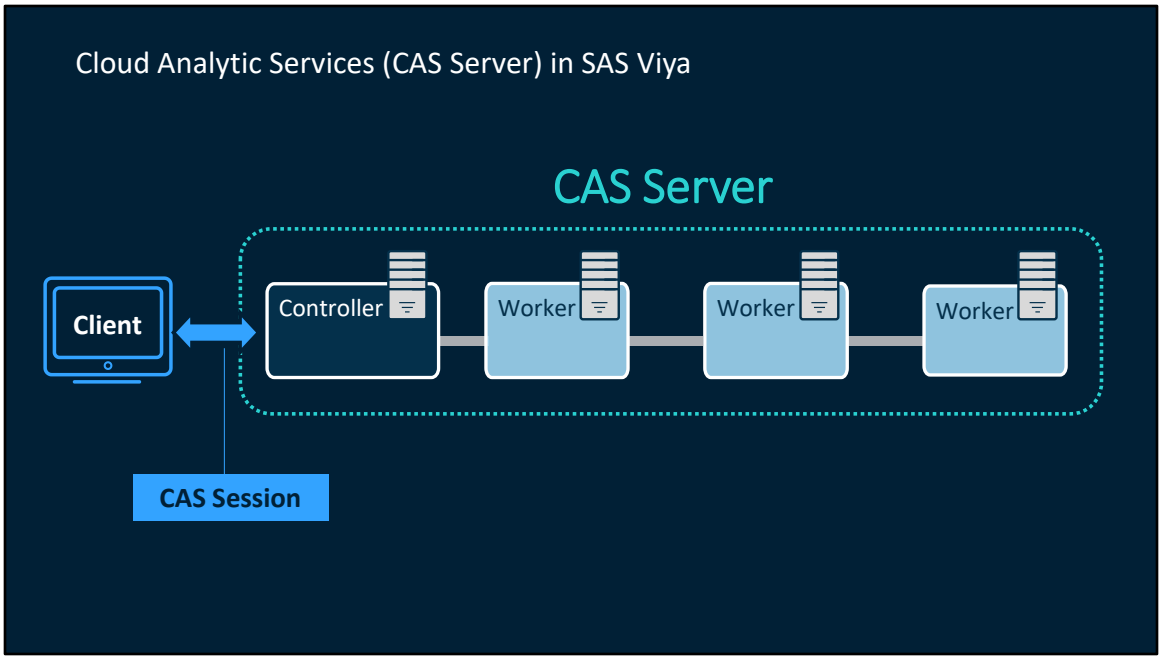


You can use the SAS Compute server as the client to the CAS server.

The CAS server is

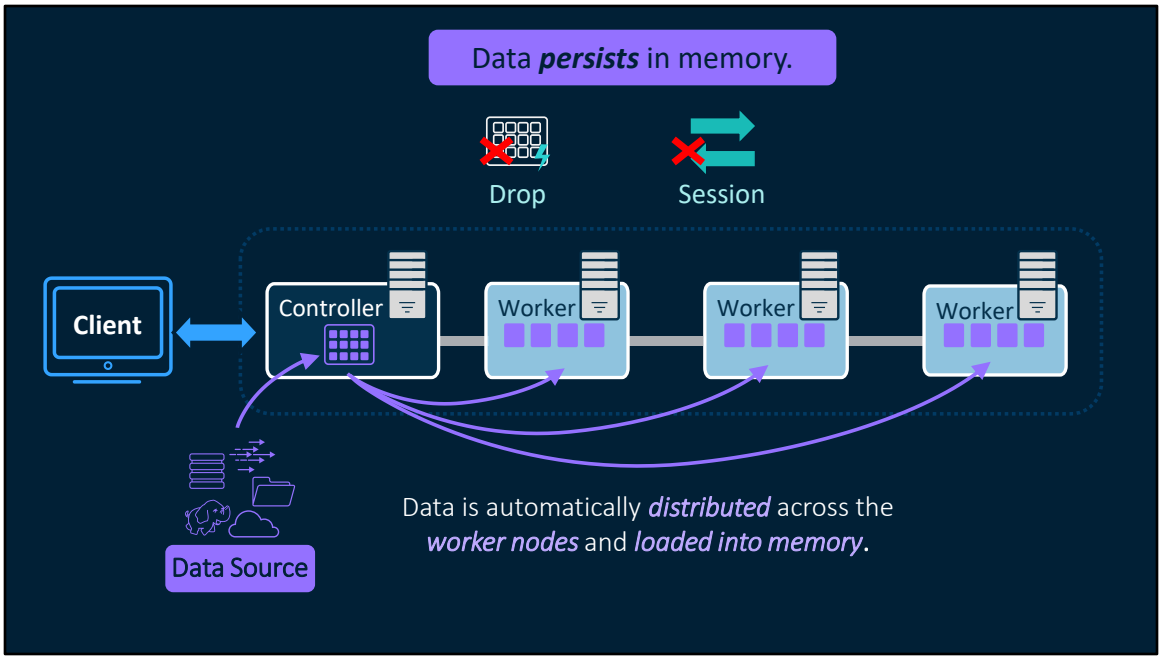
- a high performance, massively parallel environment (or MPP), built to process big data and your resource intensive programs using modern analytics and all your available SAS Viya applications like SAS Visual Analytics. It's important to know that all SAS Viya applications process data in this distributed computing environment.
- an in-memory analytics engine. Data is loaded into memory in the CAS server for parallel processing and can persists in memory for multiple reads and users until explicitly removed.
- uses SAS Viya data connectors to access data across your organization.

SAS Viya Data Connectors Documentation:
https://go.documentation.sas.com/doc/en/pgmsascdc/v_027/casref/p0j09xx6p9ffven1x7z9cq8s1zaa.htm

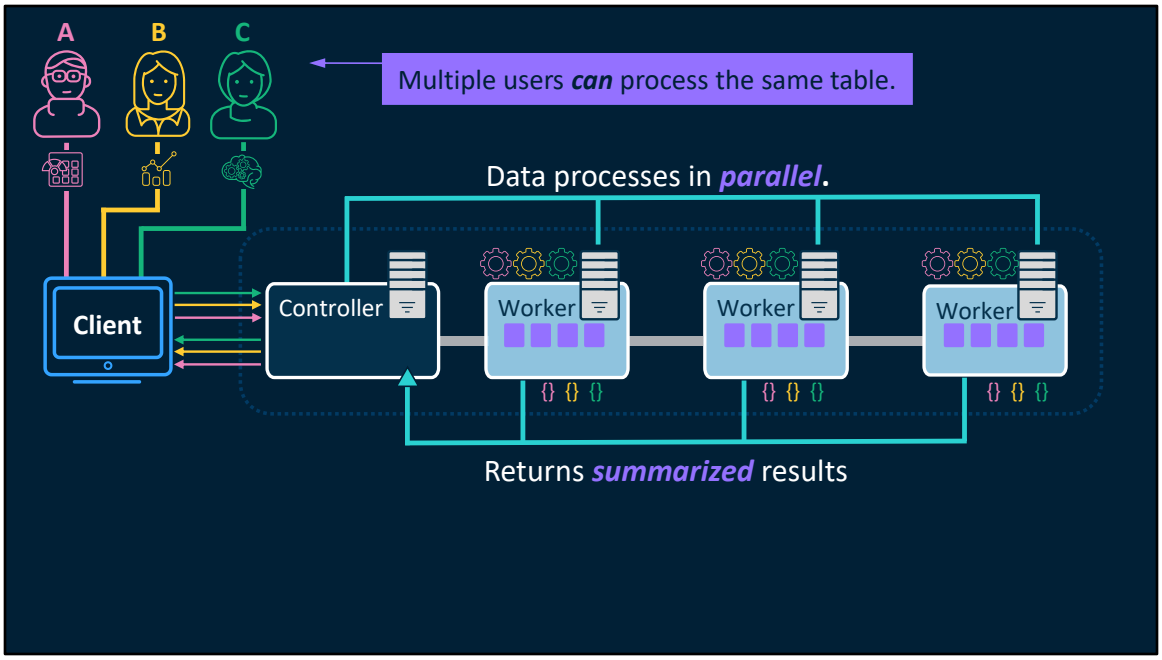


The CAS server is the cloud-native, high-performance in-memory analytics and massively parallel processing engine. The CAS server is configured to run on multiple machines. Typically, you will have one controller node and several worker nodes. You need to make a connection to the CAS server from your client. This is called a CAS session or CAS connection.

SAS Documentation: https://go.documentation.sas.com/doc/en/pgmsascdc/v_027/casfun/titlepage.htm



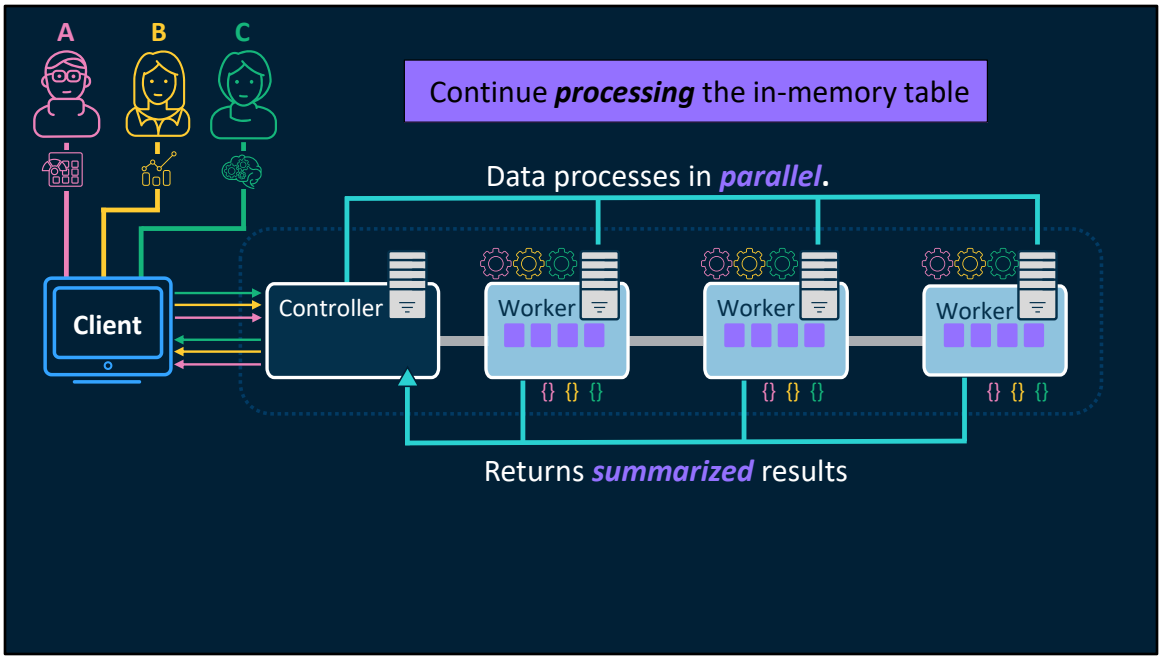
For data to be processed in CAS, data is copied into memory from some physical storage, and data blocks are automatically distributed across the worker nodes and loaded into memory. Once data is loaded into memory it persists in memory until explicitly dropped or the CAS session ends.



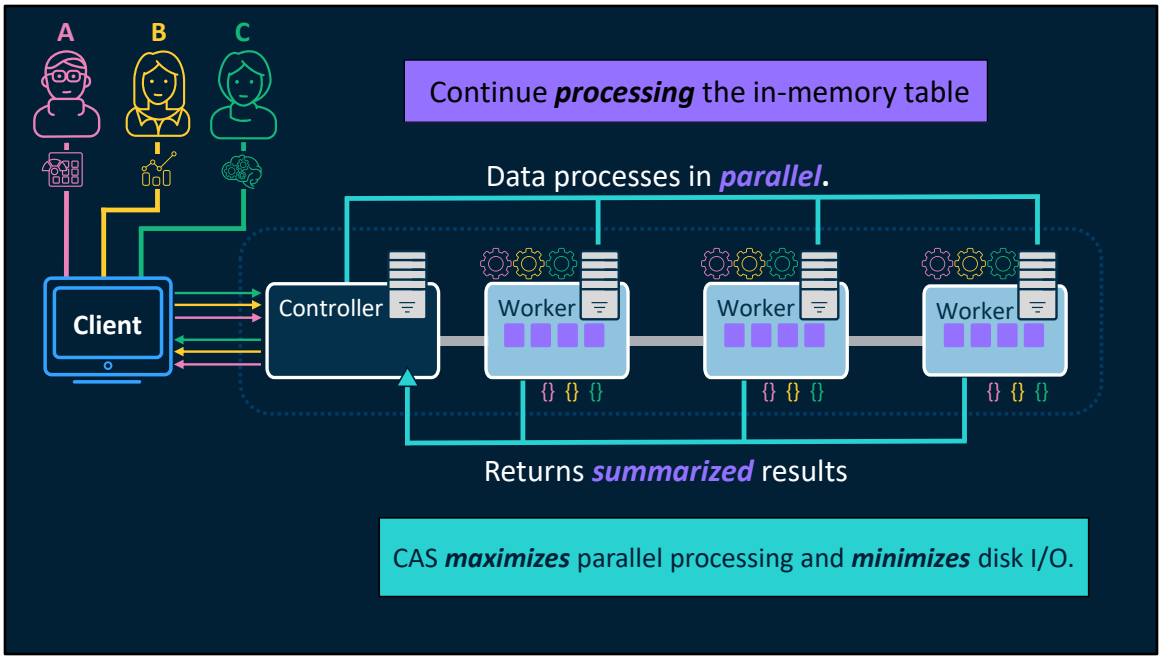
Because the data persists in memory, the same in-memory table can be accessed by multiple users and processes without any additional I/O (data load from disk to memory). Because I/O is more resource intensive, having data preloaded into memory speeds up the processing for all users.

For example, person A can execute a variety of Python programs, person B can create dashboards using SAS Visual Analytics, and person C can execute machine learning models. All users can do this using the same in-memory table copy.

In all instances, the controller accepts the programs from the client and distributes code to the workers. The workers perform coordinated parallel processing on their portion of the data, with multiple nodes executing the same actions on different parts of the data at the same time. The controller monitors progress and coordinates re-assembling the result segments produced by each worker. If requested by the client, the controller then returns the results to the client for further processing. Modifications to the in-memory table are not reflected in the physically stored data source. If you want to update the physical data source, the in-memory table must be explicitly saved back to the physical storage location.

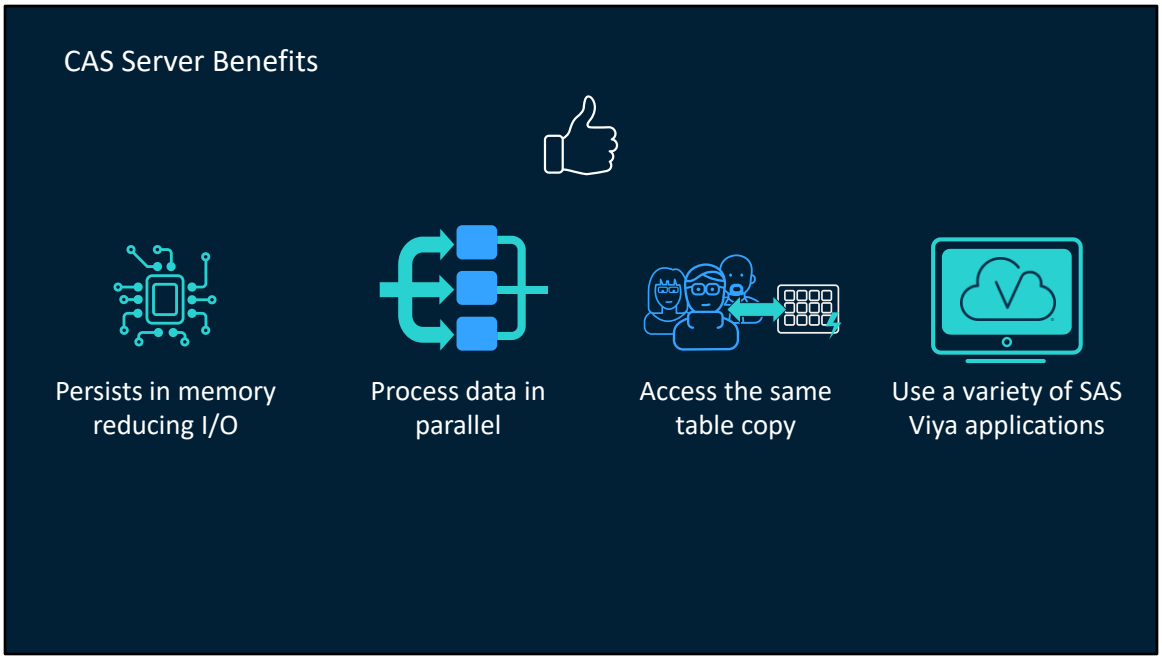


Since the data persists in memory, you can continue to process the distributed data without having to load it into memory again.



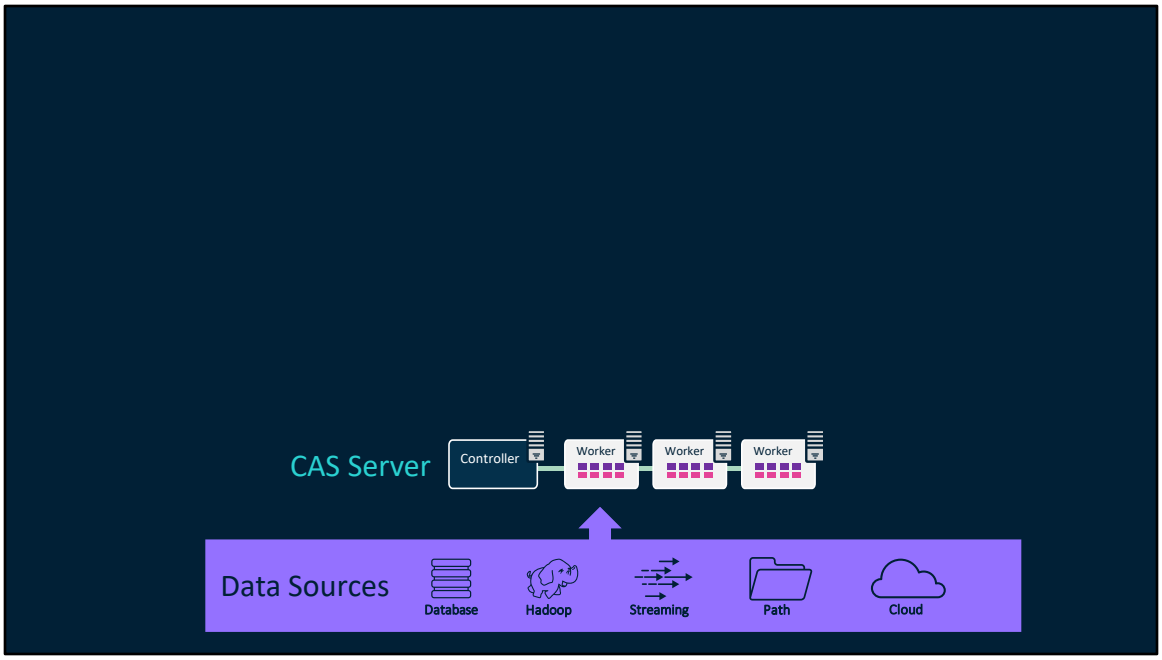
...and again.

For big data or resource-intensive programs, the CAS server maximizes parallel processing and minimizes disk I/O by reducing read/write activity and can produce results in a remarkably short period of time.

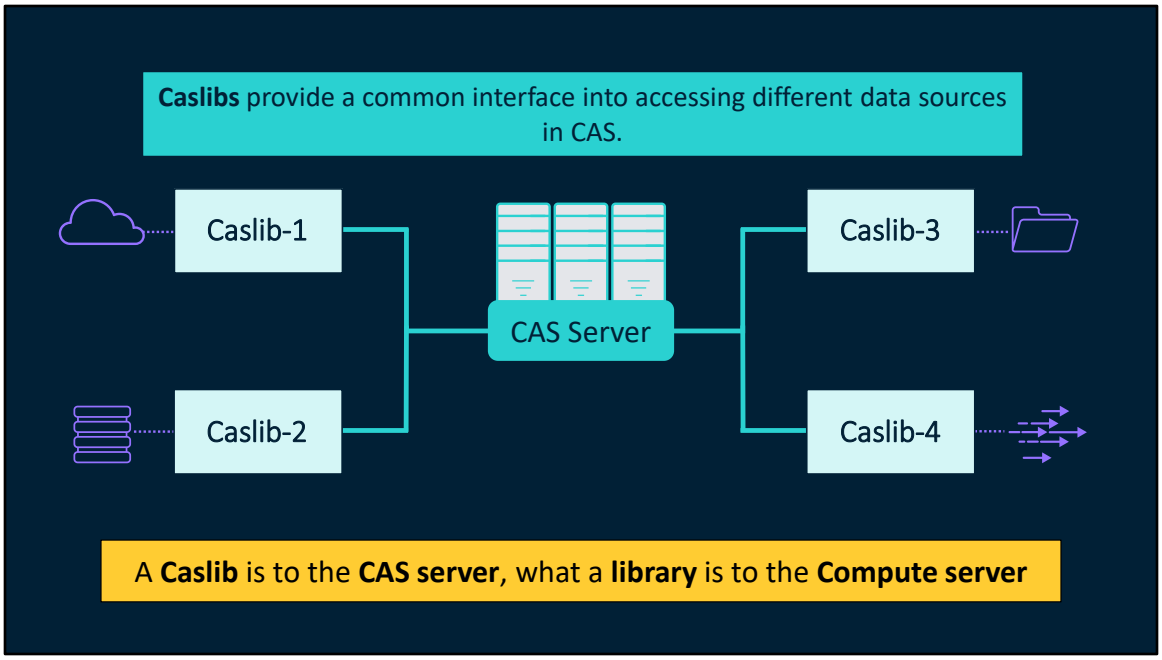


Using the CAS server in SAS Viya provides a variety of benefits.

- First, data loaded in CAS persists in memory. This reduces I/O from loading data from a physical disk into memory. Data can persist in memory for one or more users.
- Second, processing data in CAS is performed in parallel, enabling fast results on big data and resource-intensive programs like machine learning.
- Next, working with shared CAS tables avoids data duplication because users can work from the same in-memory copy.
- Lastly, SAS Viya provides a variety of applications and programming languages to process your data.



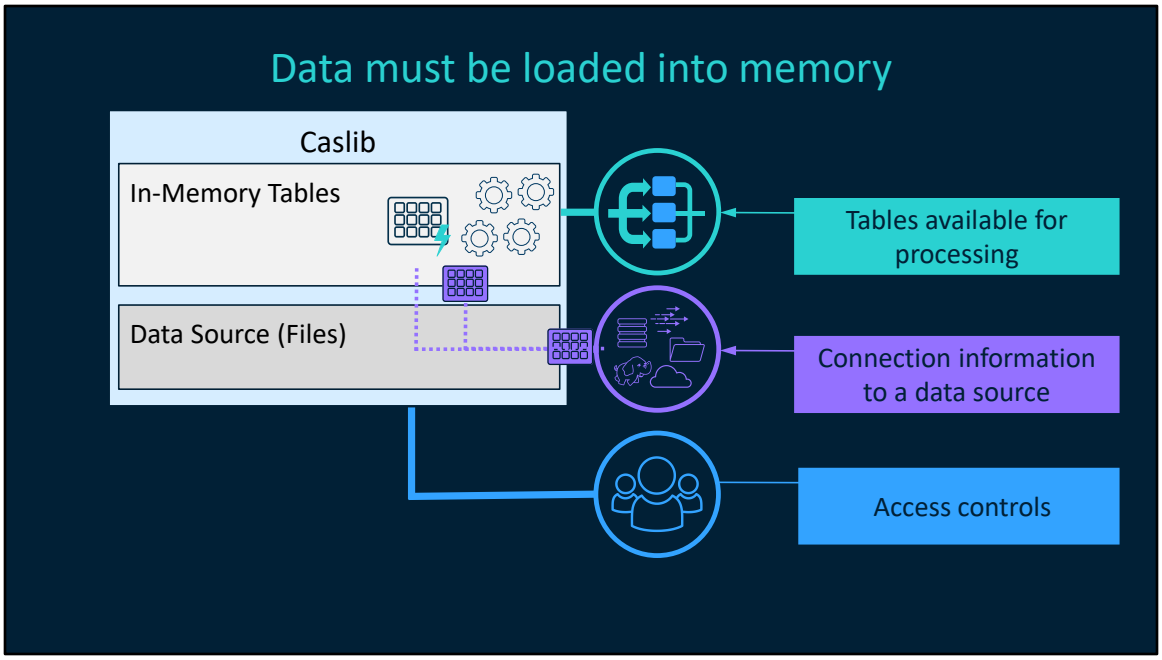
The CAS server needs to access data from some physical data source. CAS can access data from a variety of sources using SAS Viya data connectors, including databases, Hadoop, streaming data, path-based files, and data stored in the cloud.



CAS stores and accesses data via caslibs. Caslibs provide a common interface into accessing different data sources. No matter which data source you access in CAS, it is connected via a caslib. Caslibs connect into a variety of data sources in your organization.

You can think of a **Caslib** is to the **CAS server**, what a **library** is to the **compute server.**

SAS Documentation:
https://go.documentation.sas.com/doc/en/pgmsascdc/v_027/casfun/n1i11h5hggxv65n1m5i4nw9s5cli.htm

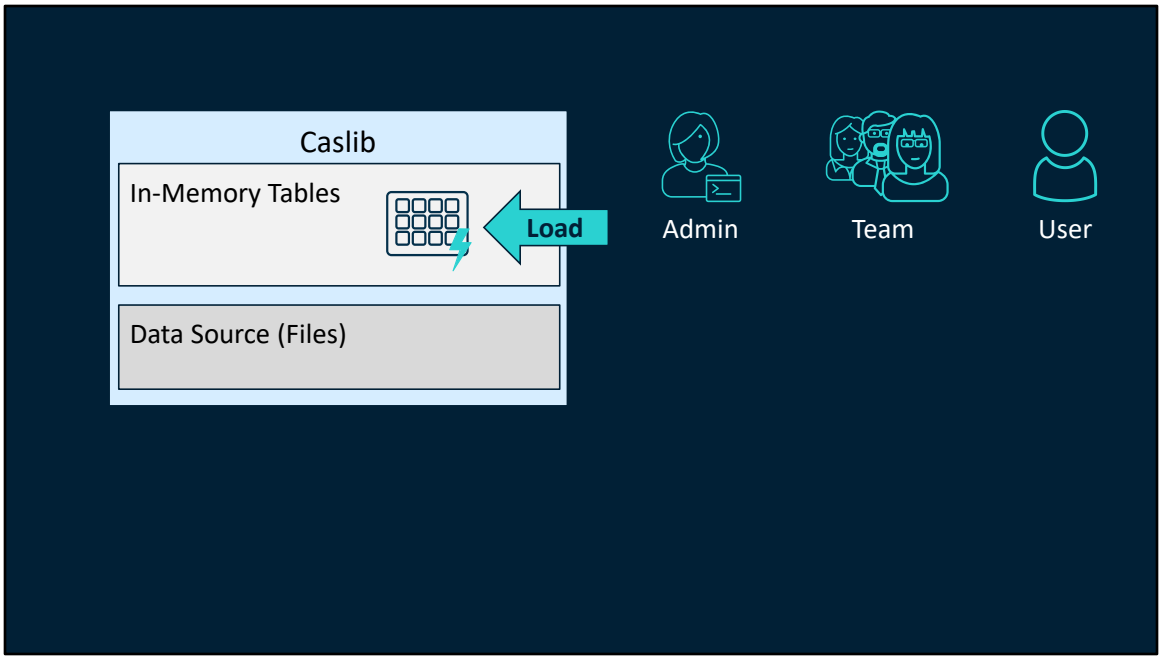


A caslib consists of three main parts.

First, there's the data source portion. The data source portion contains connection information to a physical data source for storage. Data might be stored on disk in a path with familiar formats such as SAS data sets, CSV, Microsoft Excel, or parquet files, or stored in databases, the cloud, Hadoop, or other systems. The data source portion stores the connection and authorization information required for interacting with those files using SAS Viya data connectors. Data stored in the data source portion of a caslib is generally referred to as *files*. Files in the data source area cannot be processed directly.

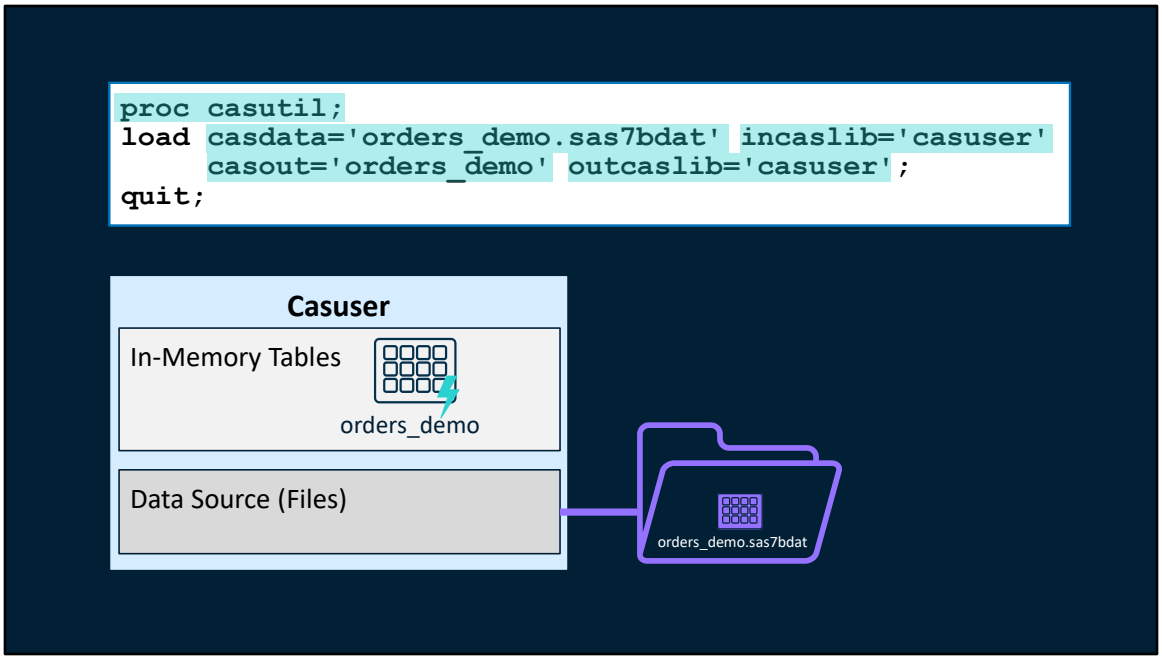
The second part of a caslib is the in-memory portion. The in-memory portion provides a space to hold tables that can have operations performed on them. You can almost think of a CAS table as a DataFrame on the CAS server. A file must first be loaded into memory from a data source. Data loaded into memory is generally referred to as *tables* or *CAS tables*. When they are loaded, in-memory tables are available for processing, and they persist in memory. Because there's no connection between the in-memory copy and the data source file, if you want to permanently keep changes made to the data during processing, the in-memory table should be explicitly written back to the data source. If you are done with the CAS table, you can drop the table. We discuss this more later.

Lastly, a caslib contains access controls about who can access that specific caslib.



Once you've identified the available caslibs and data source files, the next step is to begin processing data. To process data in CAS, the data must be loaded into memory as a CAS table. In CAS, there are various ways to load data source files as CAS tables. An administrator can load data in CAS for you, a member of your team can load data in memory for everyone, or you can load the data for yourself and others.

If you have used Python before you can think of loading data into CAS as using a read_ method in Pandas to load data from physical storage to a DataFrame. The only difference here is that you are loading data on CAS's massively parallel processing engine instead of in memory on your local machine.

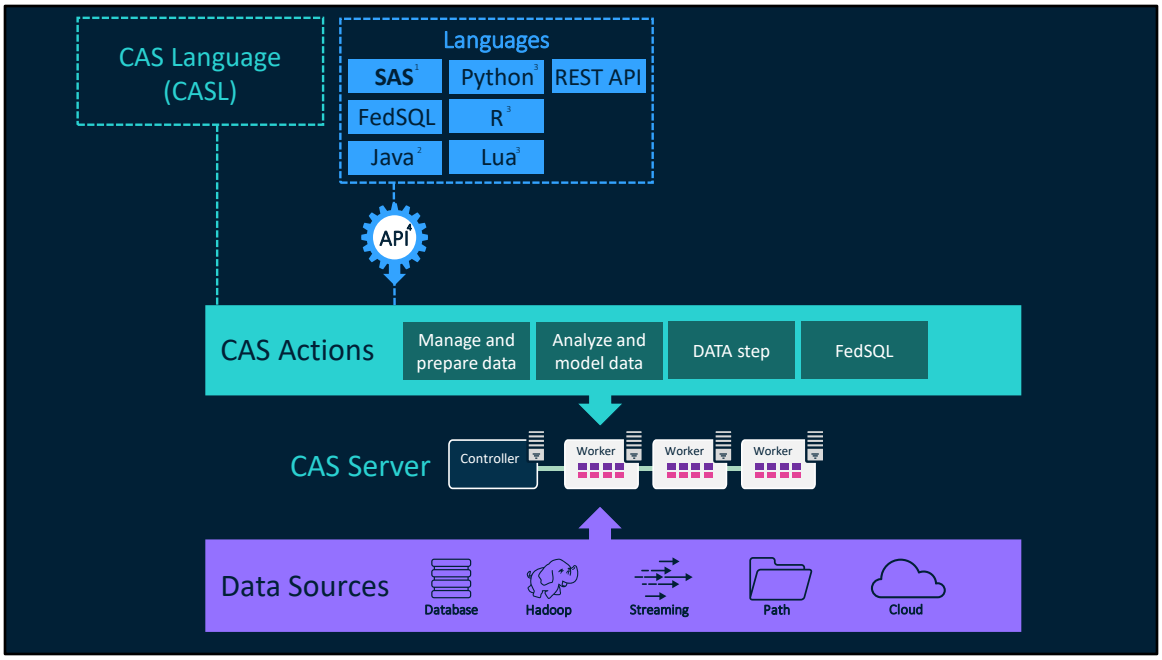


One method to load data into memory is using the CASUTIL procedure.

Within the procedure use the LOAD statement to load data into memory. Within the LOAD statement

- the CASDATA option specifies the file, the INCASLIB option specifies the input Caslib,
- the CASOUT option specifies the CAS table name, and the OUTCASLIB specifies an alternative caslib to use for the in-memory table.

SAS Documentation:
https://go.documentation.sas.com/doc/en/pgmsascdc/v_027/casref/n03spmi9ixzq5pn11lneipfwyu8b.htm



Once data is loaded into memory on the CAS server, you can process that data using CAS actions. You can think of CAS actions as the native language of the CAS server.

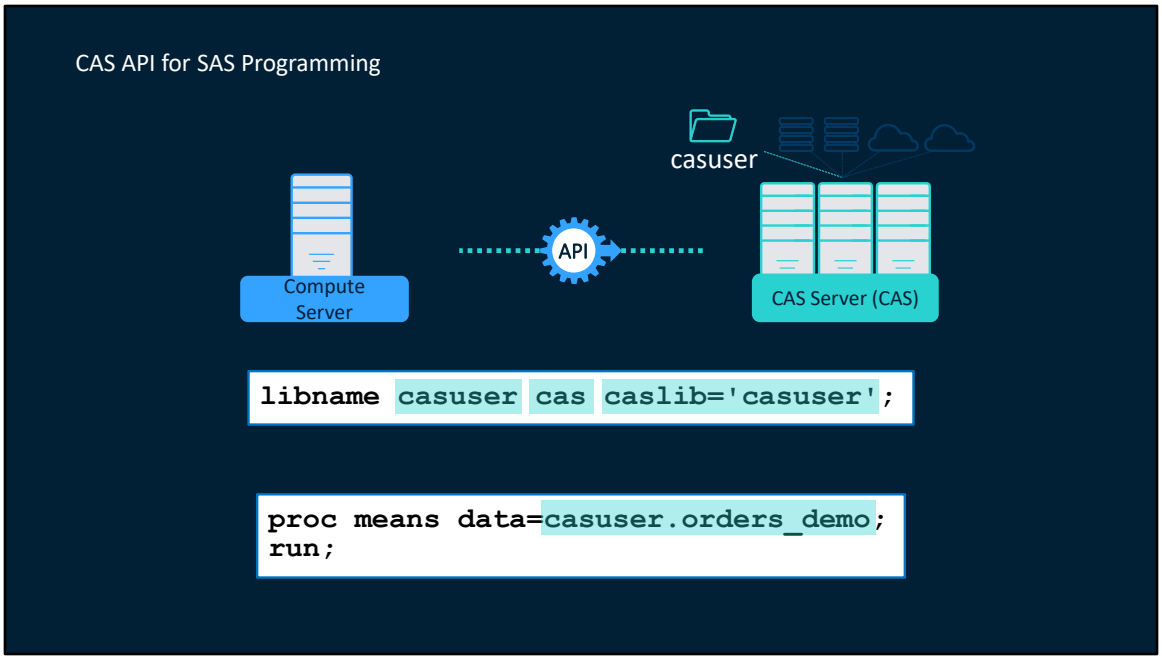
Actions are optimized units of work built for the distributed CAS server. CAS actions provide a range of functionality from managing, processing, analyzing, and modeling data, to executing most traditional SAS DATA step, FedSQL, and even DS2! The question is, how can you execute actions?

SAS Viya provides a variety of interfaces for executing CAS actions. One way to execute actions on the CAS server is to use the native CAS language, or CASL for short. CASL supports executing CAS actions on the CAS server and uses normal programing logic. If you have experience using languages like Lua, Python, or R, CASL will be an easy language to learn.

Instead of using CASL, you can also execute CAS actions through the CAS API using traditional SAS programming, FedSQL, Java, Python, R, Lua, and REST API.

NOTES:

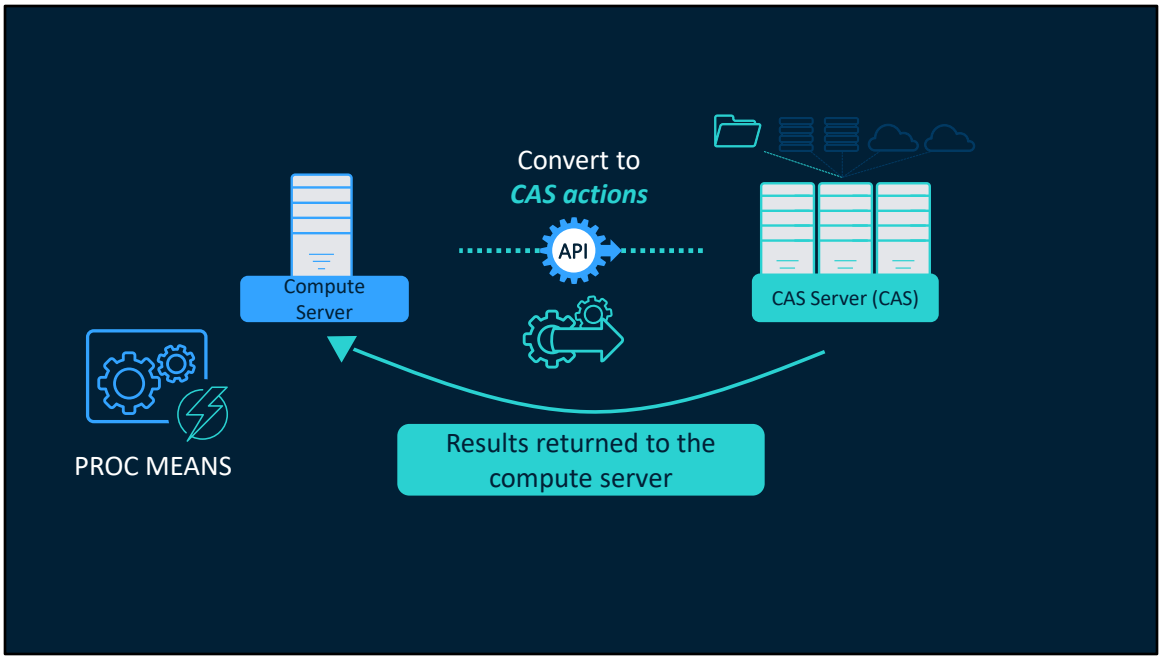
1. With the SAS programming language, many PROCs and much of the traditional DATA step is CAS enabled and can be executed on CAS tables. For more information about how to execute traditional SAS code in SAS Viya, you can view the Programming for SAS Viya course: https://support.sas.com/edu/schedules.html?crs=PGVIYA&ctry=US
2. For Java, you must use the CASClient class.
3. For Python, R, and Luna, the SWAT package is required. Many familiar methods are available through the SWAT package.
4. The CAS API converts the native language syntax to CAS actions behind the scenes.



You can use the LIBNAME statement to create a libref that points to your caslib by using the CAS engine. If you have ever used the LIBNAME statement to connect to a database, this is a similar process.

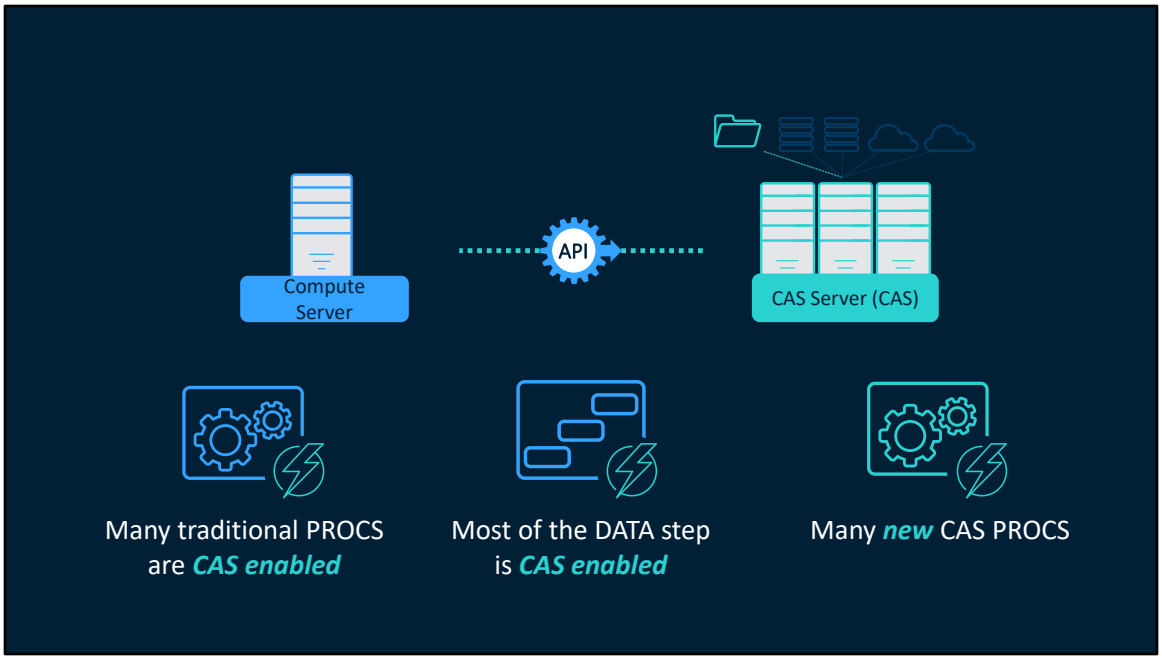
In the LIBNAME statement example, the libref **casuser** uses the CAS engine and points to the **Casuser** caslib with he CASLIB= option. After librefs are assigned to caslibs, the librefs are listed in the Libraries section and are identified by a cloud icon. After the library reference to a caslib is set, you can reference the CAS table using the traditional two-level naming convention.

NOTE: The libref can be the same as or different from the caslib name.



When using the SAS language to process data in CAS, the PROC or DATA step is converted into CAS actions behind the scenes through the CAS API using the CAS engine that was specified in the LIBNAME statement.

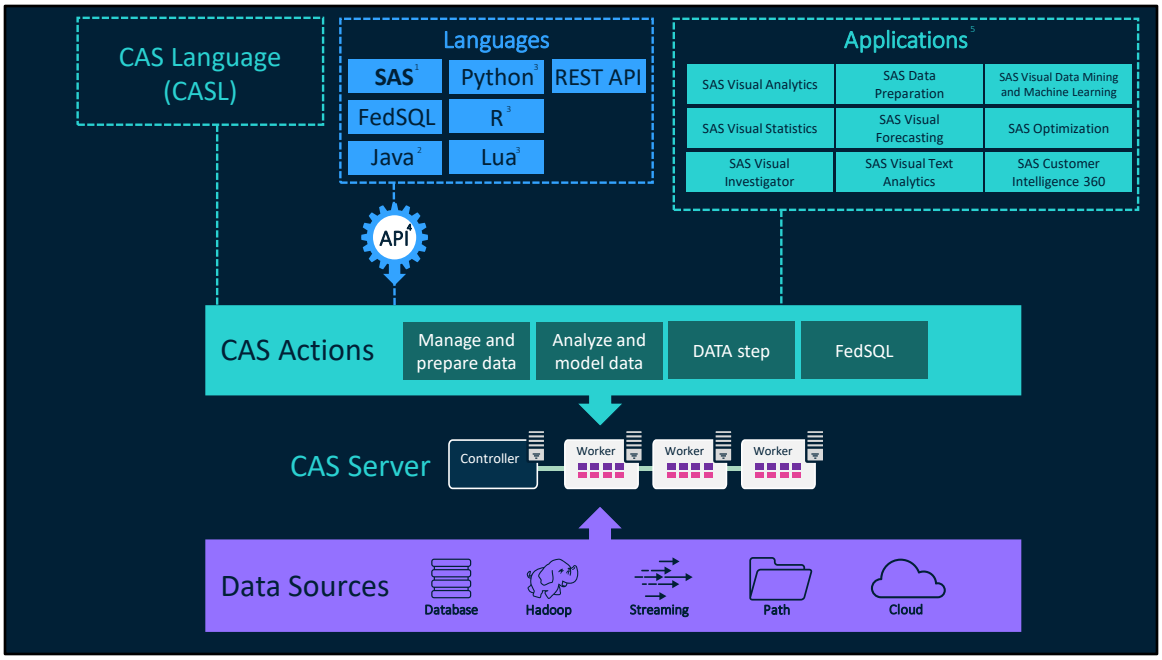
This works similarly to when processing data in a database. There you use the LIBNAME statement with a database engine to connect to tables in the database. However, when using this method, PROCS are converted to native database SQL behind the scenes instead of CAS actions.



When using the SAS programming language to process data in CAS

- many traditional PROCS are CAS enabled, allowing you to utilize the massively parallel processing engine in the CAS server with familiar SAS procedures
- much of the DATA step is CAS enabled and will process in CAS if all functions and statements are CAS enabled
- many new CAS specific PROCS were created to utilize the CAS server but have the feel of traditional SAS PROCS

SAS Documentation:
https://go.documentation.sas.com/doc/en/pgmsascdc/v_027/proc/p0nnkdmqmz48w8n1kqofzc7mcla4.htm

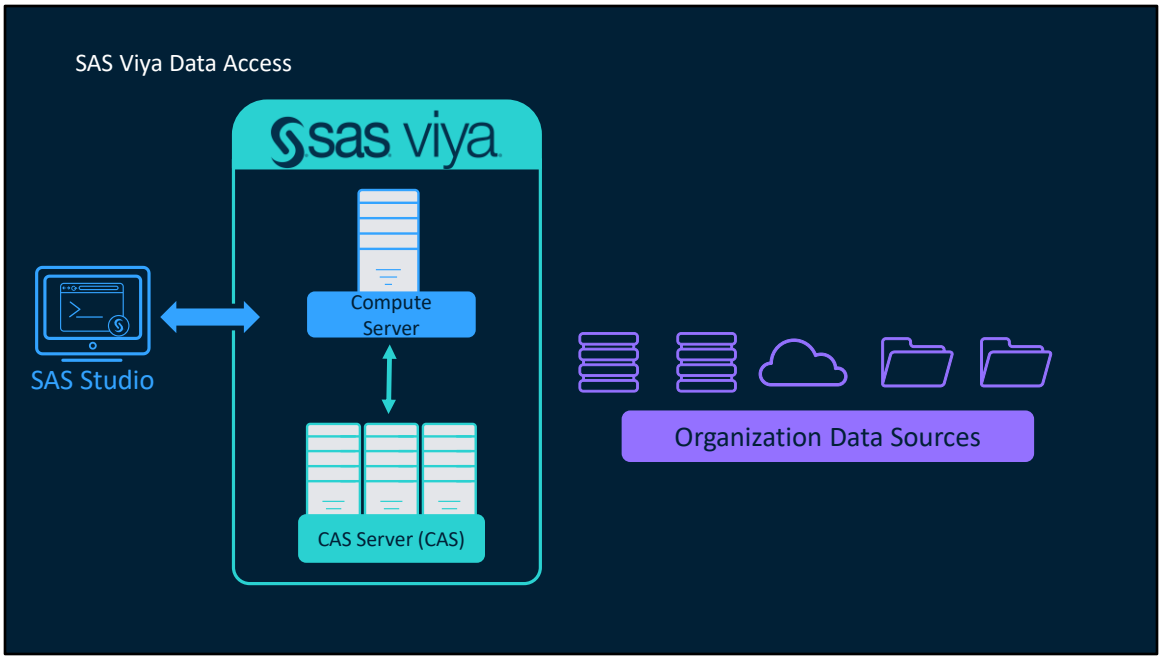


Lastly, you can use a variety of SAS Viya applications like SAS Visual Analytics to create dashboards, SAS Visual Statistics and SAS Visual Data Mining and Machine Learning for modeling and statistics, and a variety of others. All applications execute actions behind the scenes.

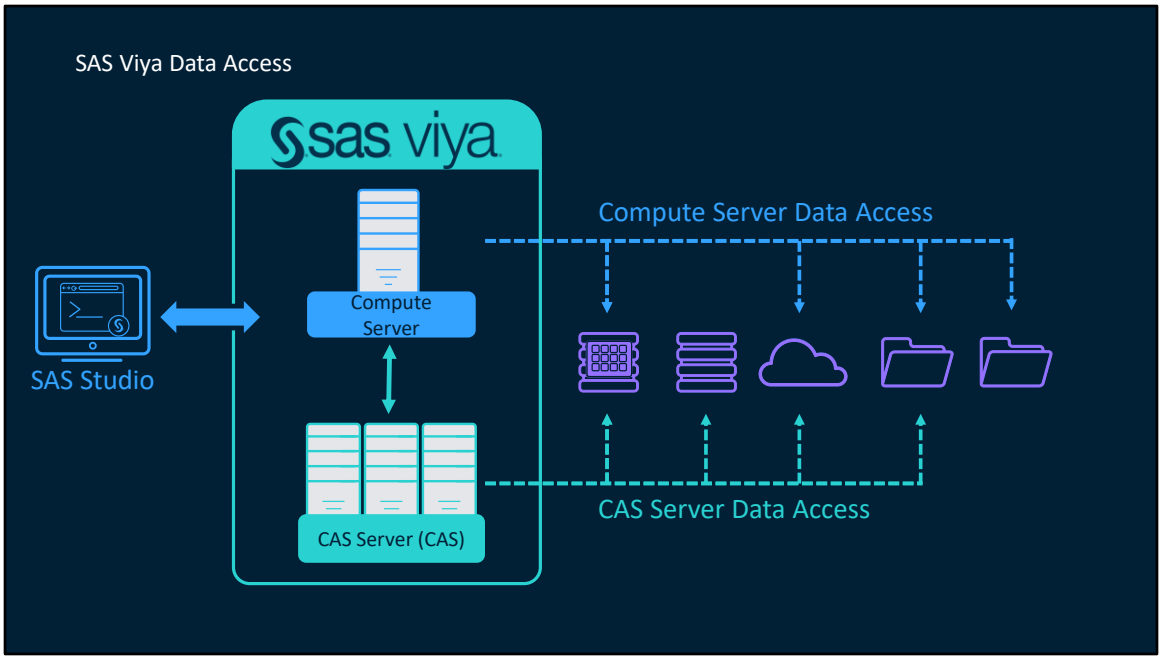
NOTES:

1. With the SAS programming language, many PROCs and much of the traditional DATA step is CAS enabled and can be executed on CAS tables. For more information about how to execute traditional SAS code in SAS Viya, you can view the Programming for SAS Viya course: https://support.sas.com/edu/schedules.html?crs=PGVIYA&ctry=US

2. For Java, you must use the CASClient class.
3. For Python, R, and Luna, the SWAT package is required. Many familiar methods are available through the SWAT package.
4. The CAS API converts the native language syntax to CAS actions behind the scenes.
5. Visit the SAS website for detailed information about all available SAS Viya offerings.



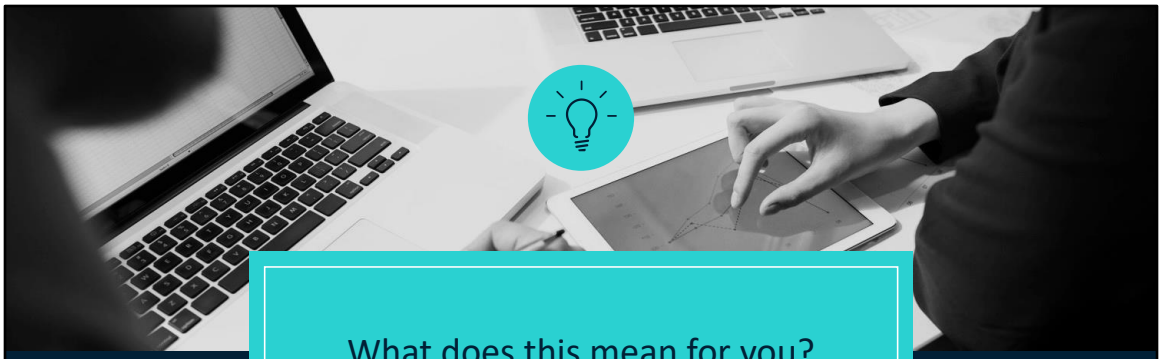
SAS Viya can access data sources throughout your organization. In this example, your organization has five data sources. Two databases, cloud data, and two folder paths.



SAS Viya can access that data using either server. In this example

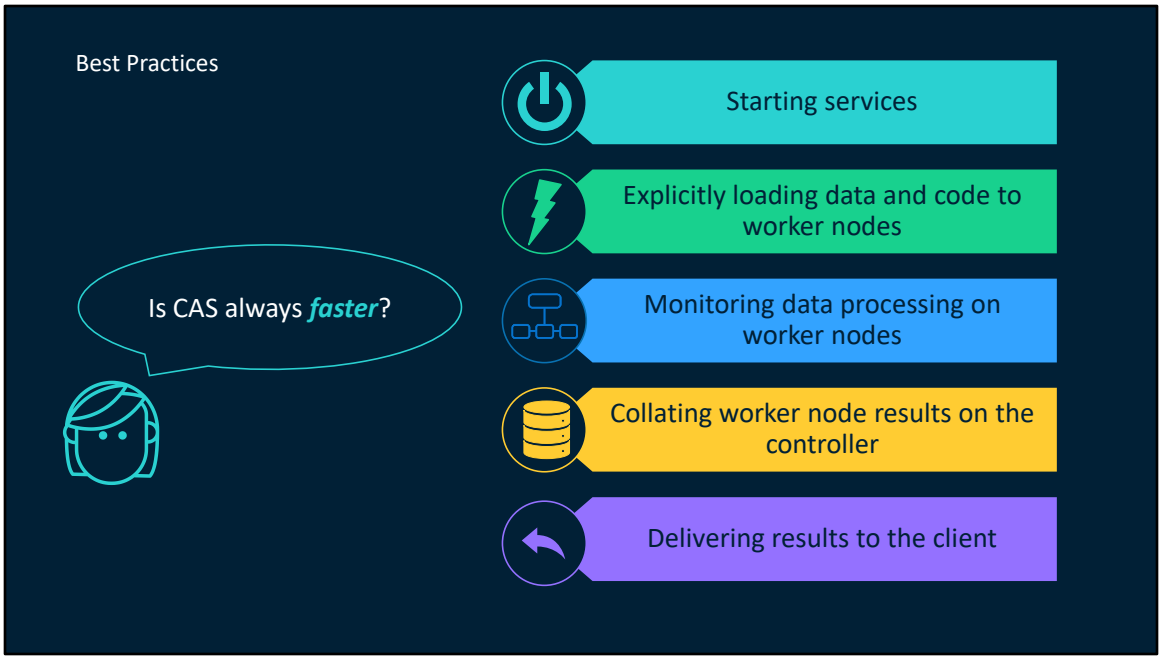
- the compute server can access some (or all) of those data sources. In this example, compute can access 4 out of the 5.
- the CAS server can also access some (or all) of those data sources. In this example, CAS can access 4 out of the 5.

Notice how both the compute server and the CAS server can access the same data sources. This means that depending on the data size and your objectives, you can read in data from the data source into the compute server, or the CAS server, depending on your needs. This is an important concept to understand, and it depends on the set up of your organization.



What does this mean for you?

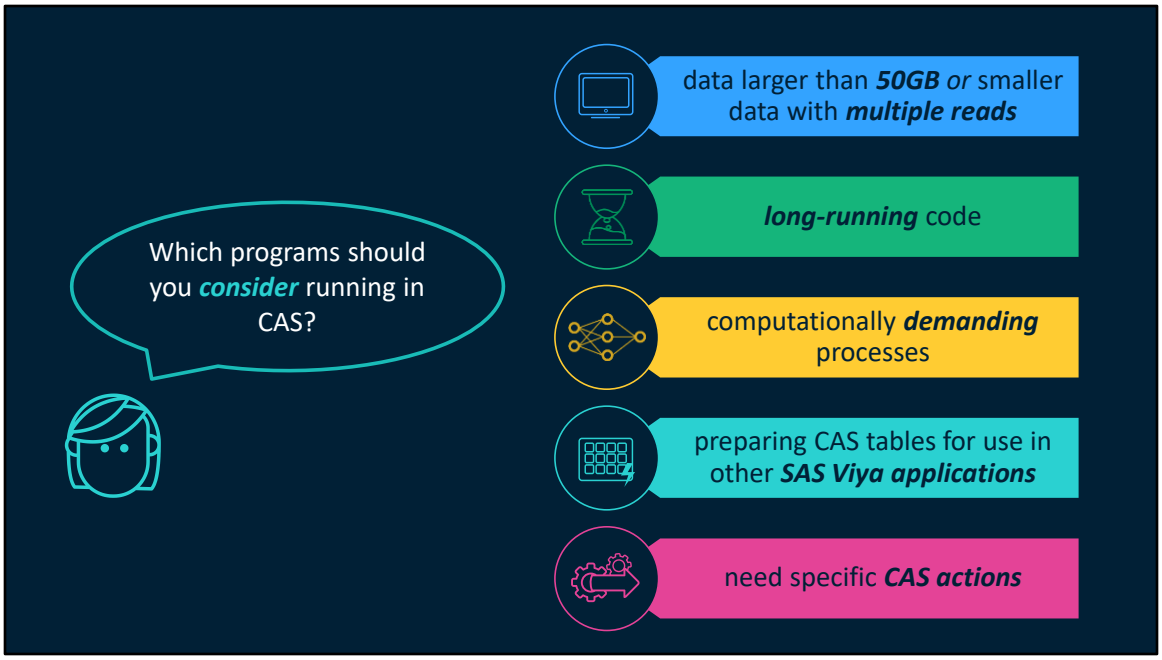
Additional resources when using the CAS server!



The first question is always, Is CAS faster than your open-source environment? Well, it depends. CAS overhead processes might include

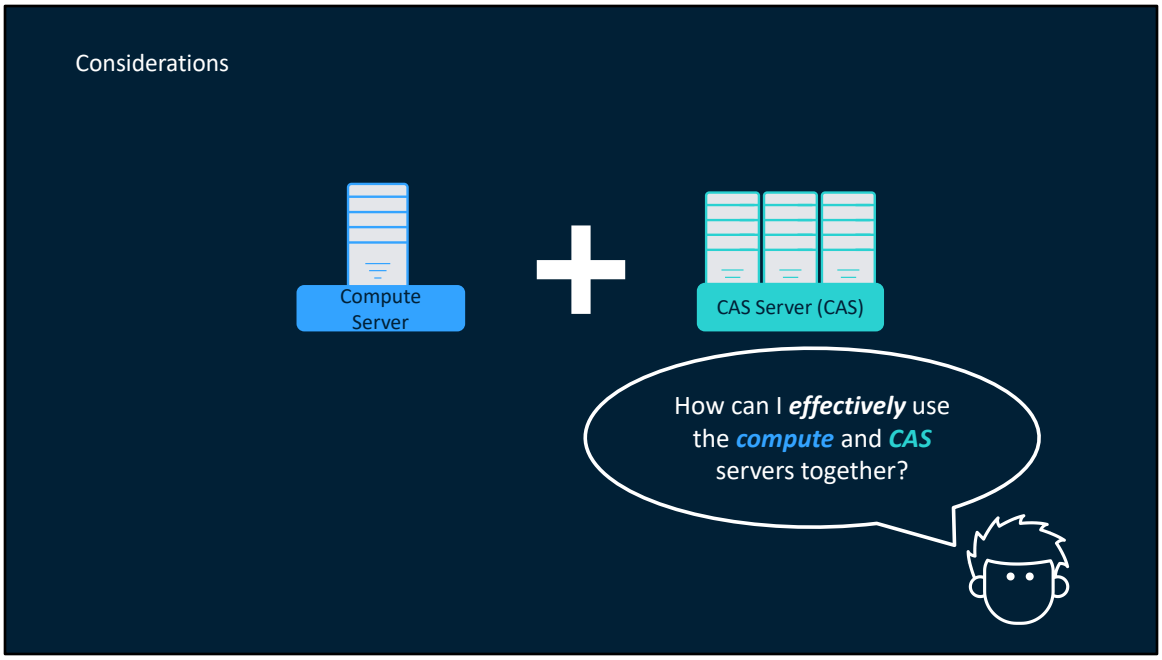
- starting the required services,
- explicitly loading data and code to the worker nodes,
- monitoring the progress as the data is processed on each worker node,
- collating results on the controller node,
- delivering final results to the client.

For small tables and simple processes, the SAS Compute Server or your open-source environment might provide quicker results.



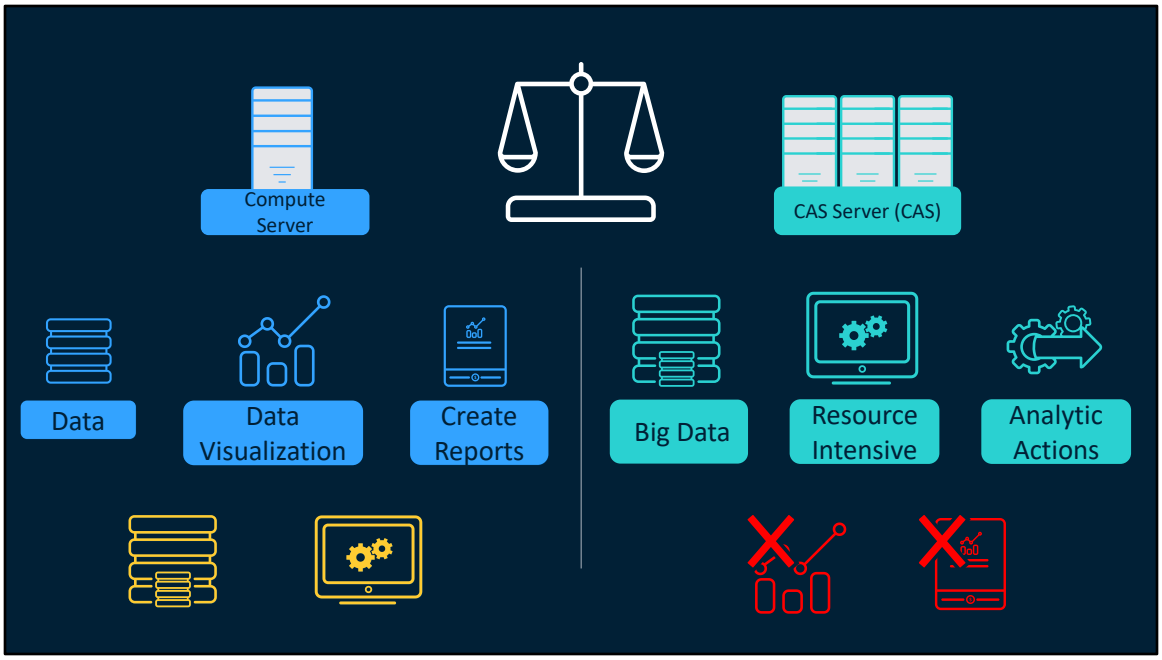
As you work in your Viya environment and consider writing or modifying programs to run in CAS, there are a few best practices to consider. CAS generally outperforms the Compute Server if you have

- data sources larger than 50 GB, or data smaller than 50GB that includes multiple reads into memory
- for code that runs for a long period of time
- PROCs that are computationally demanding, or DATA steps with extremely long or complex logic
- to prepare data for use in other SAS Viya applications
- the need for specific analytical actions that the CAS server provides



In SAS Viya, having both compute and CAS servers allows for greater flexibililty when working with big data. You can leverage the strengths of each server by understanding how to effectively use them together. The question is, how can you effectively use the two servers?

The key is using them as a team!

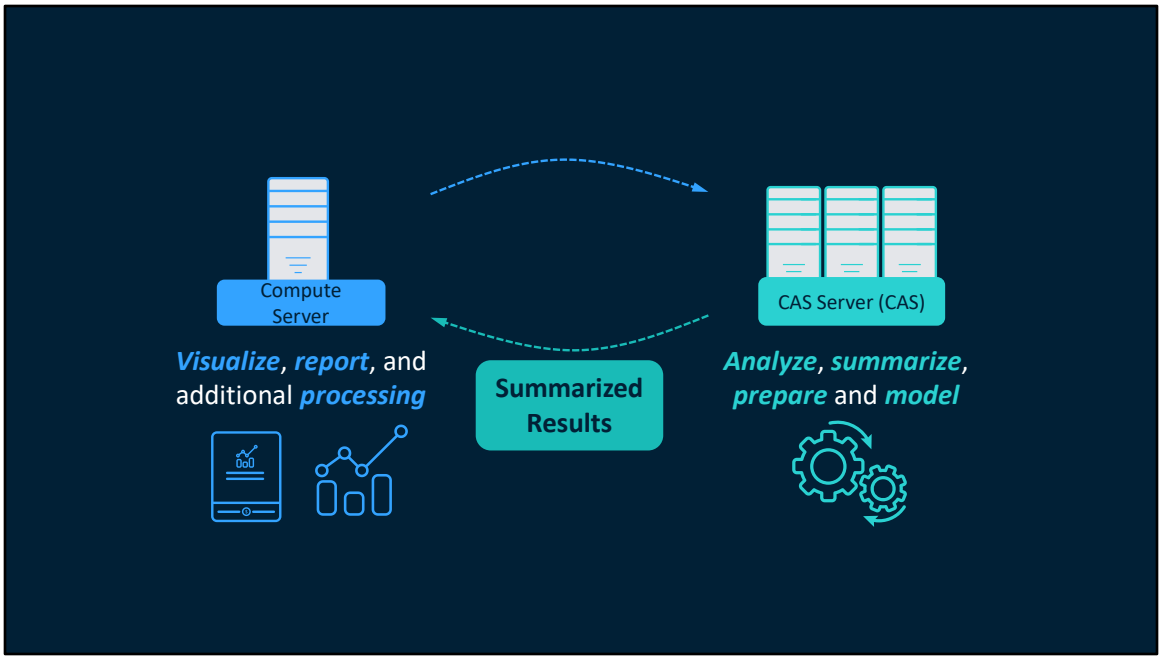


The compute server

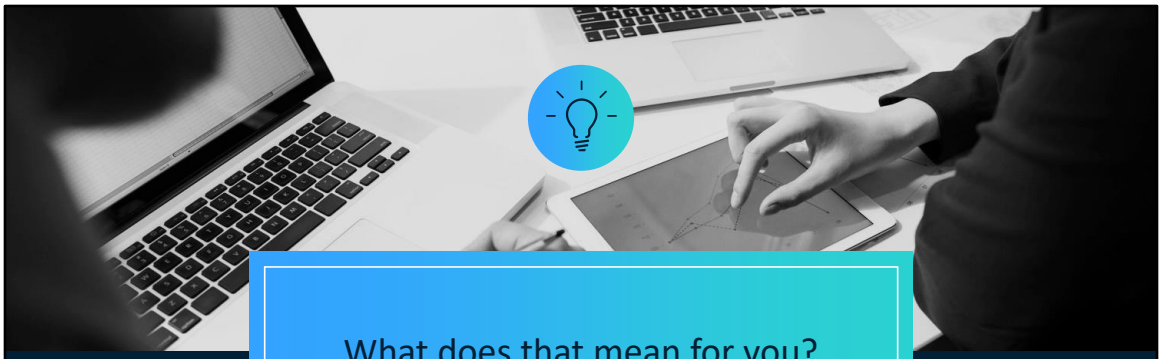
- performs exceptionally well when working with moderately sized data because it doesn't contain the startup overhead of CAS
- includes a robust data visualization and reporting toolset, which can easily produce results in a wide variety of formats, including PDF, Excel, Word, and PowerPoint. When exporting to formats like Excel and CSV, the compute server provides more robust functionality than CAS.
- has fewer CPUs and smaller memory than the CAS server, and many processes must run single-threaded. This can be less efficient when processing large data and resource intensive programs.

The CAS server

- is fantastic at working with large data, resource intensive programs, and provides additioanl analytics and data processing actions.
- cannot programmatically visualize data, or create reports in file formats such as PDF, PowerPoint, Excel, and Word.



Typically, you want to do all your , analyzing, summarizing, preparing and modeling of large data in CAS. Remember, CAS stores entire tables in memory for massively parallel processing, resulting in extremely fast results. When the processing is complete, CAS can return a subset of the data or summarized results to the compute server for additional processing, visualization, and reporting.



What does that mean for you?

Use the ***SAS compute server*** and ***CAS server*** as a team!

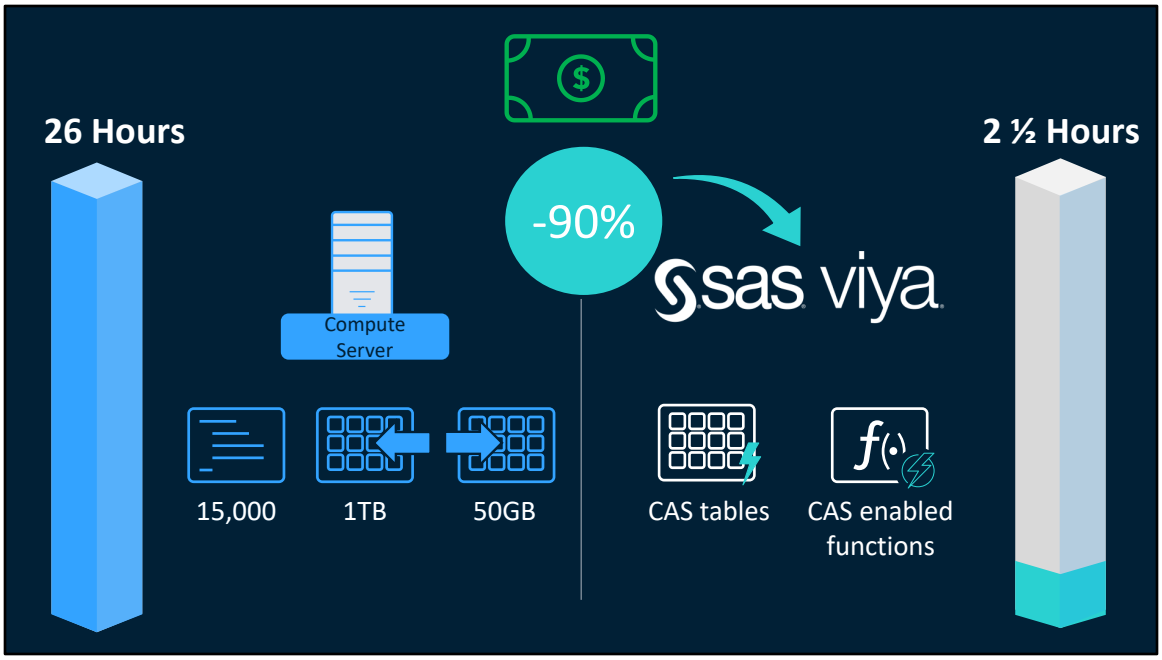
Case Studies

U.S. FINANCIAL COMPANY

U.S. GOVERNMENT

Lastly, there are two case studies I want to talk about. For more information on these case studies, you can check out the following links:

Paper: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4147-2020.pdf
Free Webinar: https://communities.sas.com/t5/Ask-the-Expert/Leveraging-SAS-Viya-to-Improve-SAS-Processes/ta-p/600509

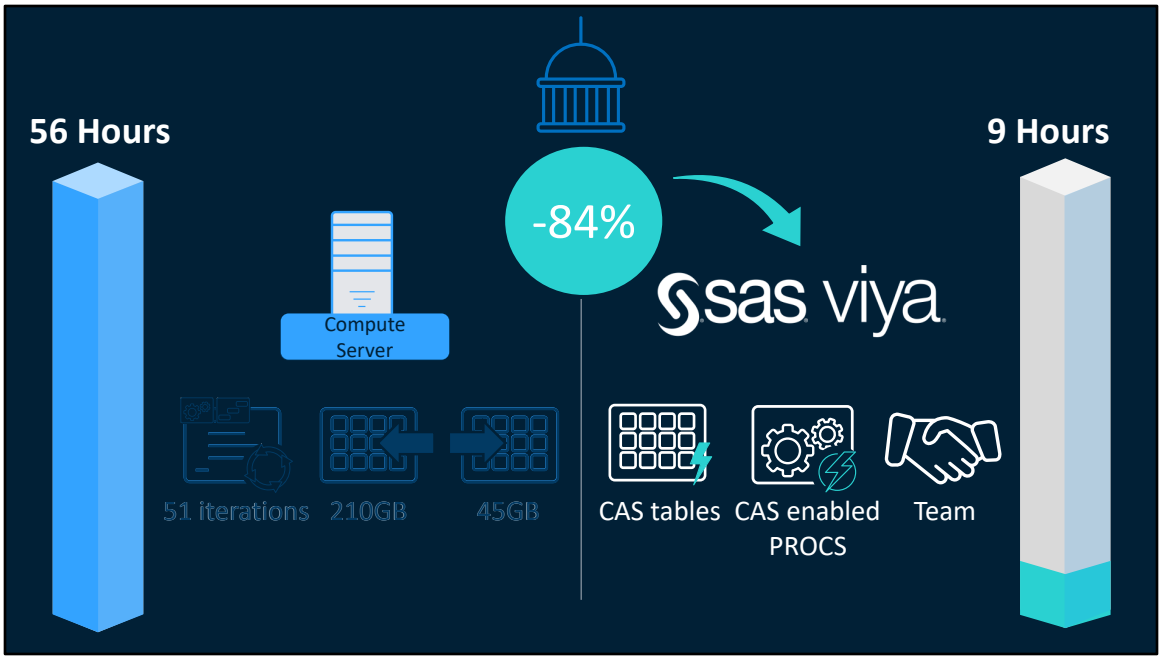


U.S. FINANCIAL COMPANY

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:
    - o Utilized PROC SORT to prepare source data for merging. Source tables resided in CAS and target tables were output to WORK in SPRE.
    - o Performed a DATA Step MERGE with source and target tables residing in WORK in SPRE.
    - o 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.

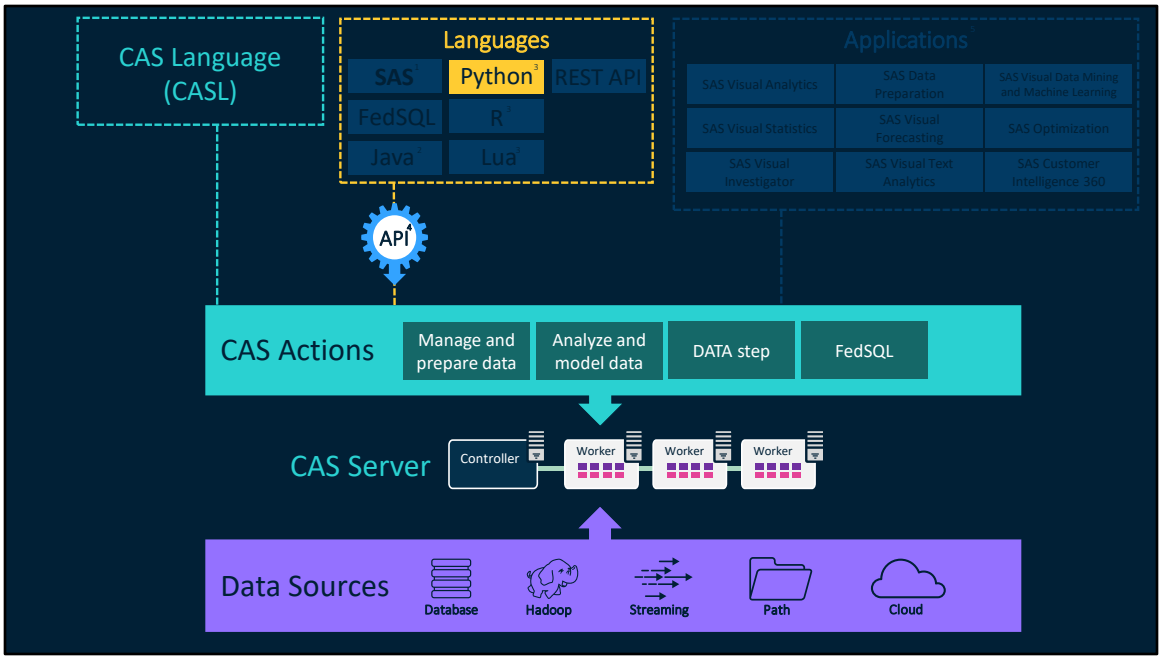
Resources

- **Leveraging SAS® Viya® to Improve SAS® Processes**
  https://communities.sas.com/t5/Ask-the-Expert/Leveraging-SAS-Viya-to-Improve-SAS-Processes/ta-p/600509

- SAS® Cloud Analytic Services: Fundamentals
  https://go.documentation.sas.com/doc/en/pgmsascdc/v_025/casfun/titlepage.htm
- CAS Action! - a series on fundamentals
  https://blogs.sas.com/content/sgf/2021/08/06/cas-action-a-series-on-fundamentals/
- Getting Started with Python Integration to SAS® Viya® - Index
  https://blogs.sas.com/content/sgf/2020/06/19/getting-started-with-python-integration-to-sas-viya-index/

Bonus Material

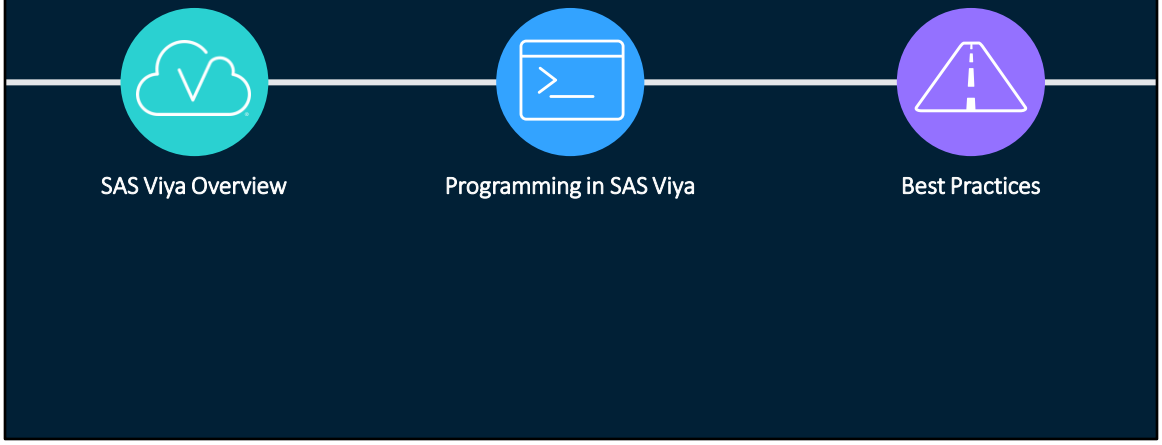
Python Integration to SAS Viya



In this presentation we used the SAS Programming languages to send process data in the CAS server through the CAS API. Instead of using the SAS Programming language, you could also use other languages like the native CAS language (CASL) or Python.

Free Ask the Expert Webinar: https://www.sas.com/en_us/webinars/use-python-in-sas-viya.html

Conclusion



Thank you for attending!

Peter.Styliadis@sas.com
Sr Technical Training Consultant

Connect with me on LinkedIn!

SAS Training

- Programming for SAS® Viya®
- SAS® Viya® and Python Integration Fundamentals
- High-Performance Data Processing with CASL in SAS® Viya®

sas.com

- Programming for SAS® Viya® - https://support.sas.com/edu/schedules.html?crs=PGVIYA&ctry=US
- High-Performance Data Processing with CASL in SAS® Viya® - https://support.sas.com/edu/schedules.html?crs=CASL&ctry=US
- SAS® Viya® and Python Integration Fundamentals – https://support.sas.com/edu/schedules.html?crs=PIVY&ctry=US

Questions?