Paper SAS1854-2018

Five Approaches for High-Performance Data Loading to the SAS® Cloud Analytic Services Server

Rob Collum, SAS Institute Inc., Cary, NC

Updated: 08 October 2018

ABSTRACT

SAS® Viya® offers the SAS® Cloud Analytic Services (CAS) Server as the in-memory analytics engine to serve the demands of powerful analytics, the scope and volume of which are ever increasing. However, before that analysis can occur, large quantities of data must often be loaded into the CAS memory space. Therefore, the transfer of huge amounts of data from source systems into CAS is a major consideration when planning the architecture of your SAS solution.

High-performance data transfer is often provided by loading data using multiple parallel channels. This process uses many pathways simultaneously to transfer data at a rate several times faster than is typical using a single, serial path. High-performance data transfer is a natural complement to the speed and efficiency of in-memory analytics provided by CAS. CAS now provides five different major techniques for the parallel transfer of data over a wide range of potential data sources.

This paper illustrates those techniques, explains the technology, and highlights the benefits of high-performance data transfer across multiple parallel channels.

INTRODUCTION

The SAS® Cloud Analytic Services (CAS) Server can reach new heights in the pursuit to process ever-growing volumes of data faster than ever. By leveraging massively parallel processing (MPP) concepts, we can scale up the analytic power of a CAS Server by adding more host machines to the cluster. In this way, large analytic problems can be broken down into chunks that are processed by CAS simultaneously across many machines.

This approach for tackling analytic problems by breaking them into smaller parts processed in parallel can also be applied to the challenge of transferring large volumes of data into CAS. The faster we're able to load data into CAS, then the sooner our users are able to investigate that data, discover new information, and share it with others.

PARALLEL VERSUS SERIAL

CAS can grow in processing capacity to meet the demands of the largest data processing challenges. If CAS was deployed to run on a single host machine, then its processing power would be limited by that machine's maximum size in terms of RAM and CPU.

Figure 1 shows CAS limited to the maximum size of a single host machine.

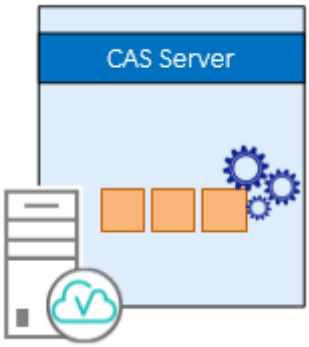

Figure 1. SAS Cloud Analytics Server Processing Data on a Single Host Machine

CAS certainly can run in a single-server mode. This is useful mainly for situations with relatively small volumes of data and few concurrent users. However, CAS was built from the ground up to take on the largest problems so we can deploy CAS in MPP mode when we're ready for the big jobs.

SAS has built massively parallel processing (MPP) concepts into CAS. Therefore, we can deploy the CAS software to run across multiple host machines and yet still act as a single logical server for analytic processing. Scalability in this way is no longer limited by the maximum specifications of a single host, but is achieved instead by adding more hosts to the cluster running CAS software.

In Figure 2, CAS is deployed to run as a single logical service hosted on many machines.

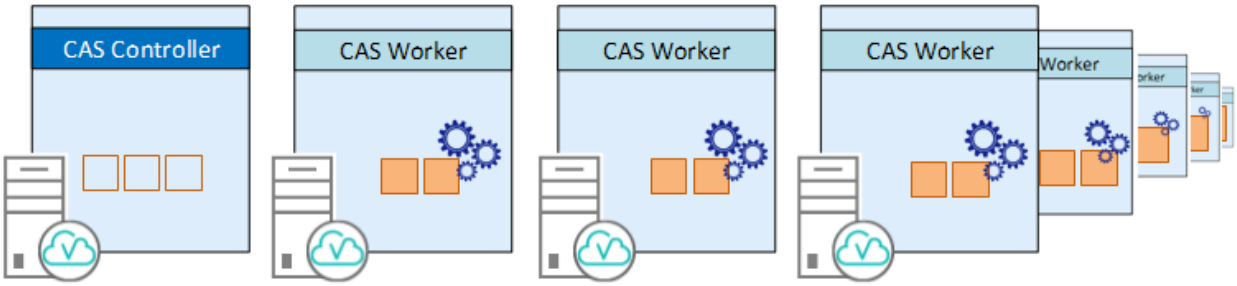

Figure 2. SAS Cloud Analytics Server Processing Data on Many Machines Simultaneously

With multiple host machines running CAS software, the conventional approach for loading data in for processing via a single stream is no longer efficient. And with the data volumes we aim to process driving us to the MPP paradigm, we need to look at using concurrent data streams for loading data into CAS more quickly and efficiently.

SAS terminology typically refers to serial loading of a CAS Server in MPP mode when all of the data is sent through the CAS controller which then distributes the data evenly across its associated CAS workers.

Figure 3 illustrates data that is transferred serially through the CAS controller.

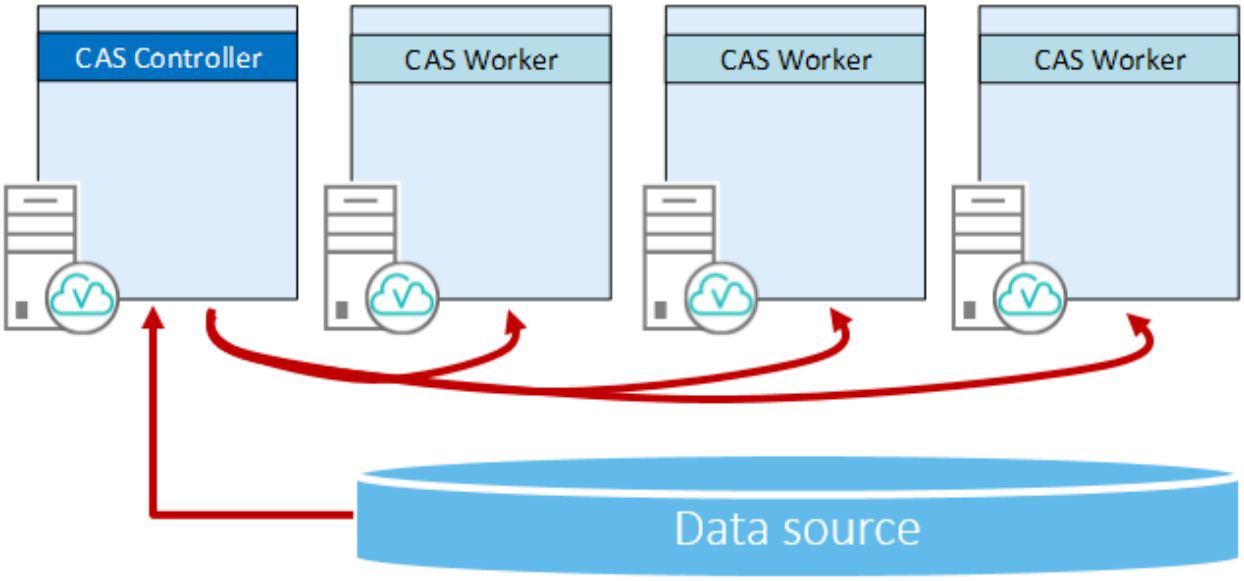

Figure 3. Serial Loading from Source to the CAS Controller That Distributes Data to the CAS Workers

Note, however, that SAS terminology aside, data transfer can still be limited to a single, effectively serial path by the environment itself. If a source table is saved as a file on single hard disk, then multiple readers of that file won't speed things up significantly – the competition for that single resource might even slow overall response as compared to servicing just one individual request for the file.

SAS has pioneered parallel approaches to data transfer for years, even before the SAS in-memory analytics products first debuted. The SAS Scalable Performance Data Server (SPD Server) is an example of a data storage technology that enabled multiple, concurrent I/O pathways for feeding data into Base SAS. Much of that technology eventually found its way into Foundation SAS as the SPD *Engine.*

When parallel transfer is implemented for the CAS Server, each of the CAS workers directly participates in data transfer, usually with some coordination by the CAS controller. The total I/O throughput rate can be multiple times the maximum possible over serial using the same hardware.

Figure 4 shows the CAS controller coordinating the transfer of data from source to each CAS worker.

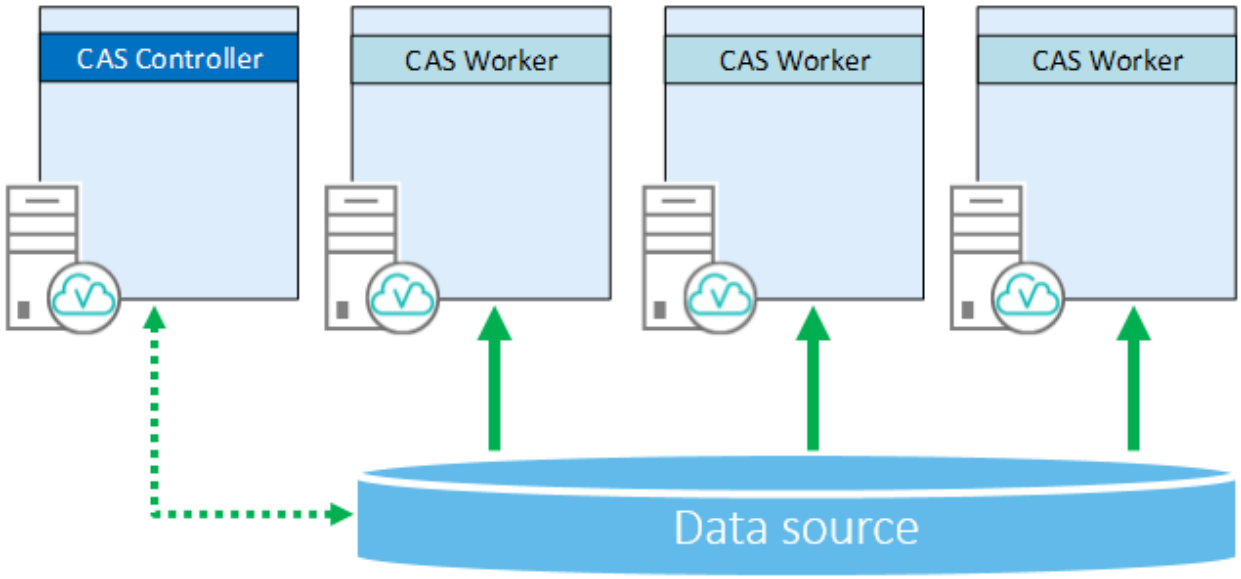

Figure 4. Parallel Loading from Source Directly to the CAS Workers as Coordinated by the CAS Controller

Keep in mind that although the SAS software might perform parallel data transfer successfully, it's ultimately up to the hardware infrastructure to do its part to provide concurrent pathways for that parallel data transfer. There might be aspects of the infrastructure, unknown to SAS, that impede efficient data transfer.

Serial loading offers the widest range of possible data sources for the CAS server. SAS data sets are one type of data source for CAS. This means that any data source that SAS 9 can read is ultimately available to CAS. SAS Viya introduces SAS Data Connectors which facilitate data transfer to and from the CAS server. Data connectors are analogous in concept to the SAS/ACCESS products for Base SAS.

SAS Viya also introduces SAS Data Connect Accelerators for CAS. A data connect accelerator works directly with the SAS In-Database Embedded Process to perform accelerated data transfer to CAS. Of course, the EP can be used for much more than simply feeding data to CAS, providing in-database capabilities like scoring, data quality, queries, code execution, and more.

Beyond the data connectors and data connect accelerators, CAS also has the ability to work directly with native data formats such as SASHDAT, SAS data sets, text-delimited files (CSV), event streams and more.

№ 1: SASHDAT ON HDFS

Our first high-performance data loading technique is an oldie but a goody. If you've worked with the SAS LASR Analytic Server and the Hadoop Distributed File System (HDFS) in the past, then you might be familiar with the SASHDAT format. SASHDAT is a SAS data storage structure that is designed specifically for high-performance, concurrent data access. With SASHDAT, your data is broken down into blocks. When saved to HDFS, those blocks are distributed across multiple hosts. When CAS loads SASHDAT from HDFS, the blocks that make up your table are read from the Hadoop DataNode hosts and re-assembled into a single logical table in the CAS memory space. The benefit of using SASHDAT is that it is often the fastest, most-efficient way to (re-)load data into CAS.

For LASR, SASHDAT requires that the LASR root and worker nodes are placed symmetrically alongside the NameNode and DataNodes, respectively, of a supported distribution of HDFS. The SASHDAT table is accessed directly on disk as blocks in HDFS (using a technique known as short-circuit read and write).

We rely on HDFS to automatically manage the availability of those blocks of SASHDAT for failover and performance.

We can use this same architecture approach with CAS, too. When we do, data stored in SASHDAT is lifted within each host from the HDFS disk directly to RAM, without any data transfer on the network between hosts.

In Figure 5, CAS is symmetrically co-located alongside HDFS to directly lift SASHDAT blocks from direct-attached disk into memory.

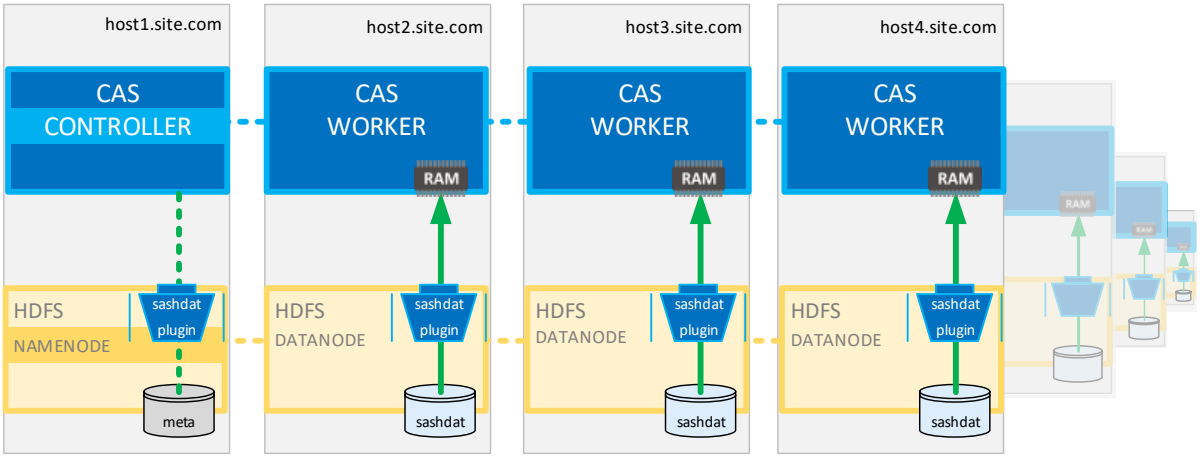

Figure 5. CAS Loads SASHDAT from Symmetrically Co-located HDFS

But CAS can go beyond what LASR was capable of. Now with CAS, we can also work with SASHDAT tables that are stored in a remote HDFS cluster. We no longer need to require symmetric co-location with HDFS to take advantage of SASHDAT. CAS uses SSH (secure socket shell, an encrypted network communication protocol) to connect with remote Hadoop DataNode hosts and fetch SASHDAT from HDFS.

Figure 6 illustrates the separation of CAS from HDFS where data is transferred across the network between machines.

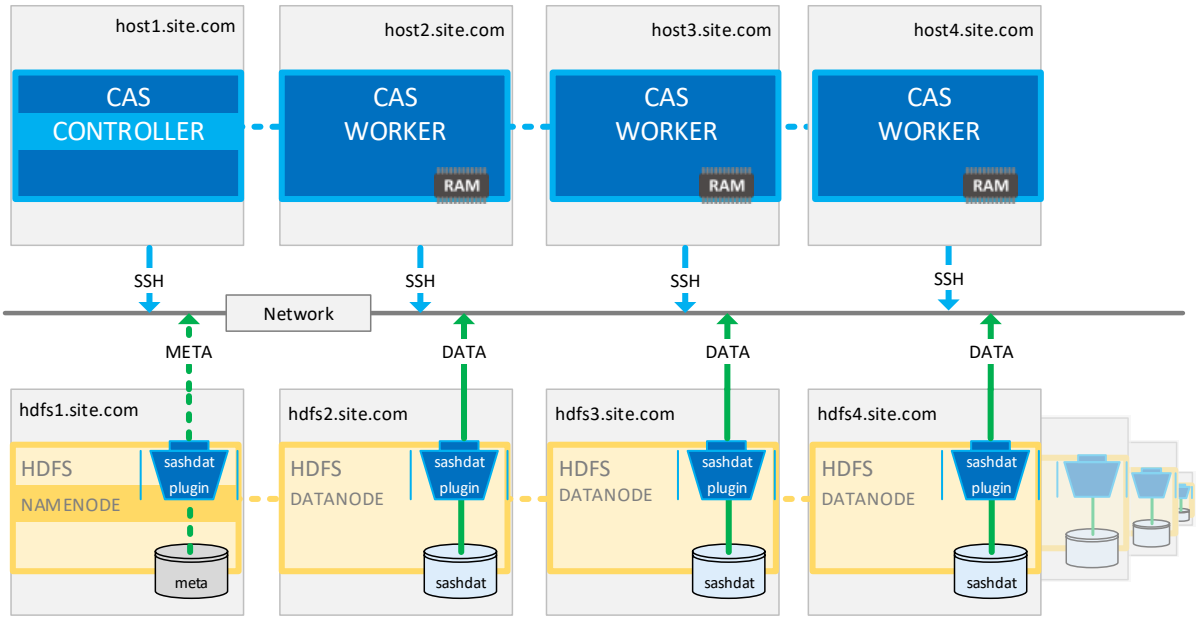

Figure 6. CAS Loads SASHDAT from Remote HDFS over the Network

The primary benefit of this separation is flexibility because the CAS and Hadoop clusters can be managed independently. The biggest drawback is that this approach isn't likely to be as fast at transferring data when compared to CAS being placed symmetrically co-located with HDFS on the same hosts. Take care to plan for the anticipated load on your network as well.

Of course, these and other tradeoffs should be weighed in consideration of your site's requirements.

Table 1. High Performance Data Transfer Attributes of CAS with SASHDAT in Hadoop

Coordination	CAS controller directs workers
Caslib srctype	HDFS
Format	SASHDAT blocks, CSV, SPD Engine
Storage	Hadoop Distributed File System
Additional software required	SAS approved HDFS

№ 2: SAS DATA CONNECT ACCELERATORS WITH THE SAS IN-DATABASE EMBEDDED PROCESS

What if you have large volumes of data in a database management system (DBMS) for which SAS offers its in-database technology? In this case, we can use the SAS In-Database Embedded Process in supported remote data providers to perform parallel loading of data directly to the CAS workers.

For the SAS Viya platform, CAS can communicate with the EP by using its new SAS Data Connect Accelerator technology.

Figure 7 shows the EP transferring data from each DBMS node to all of the CAS workers.

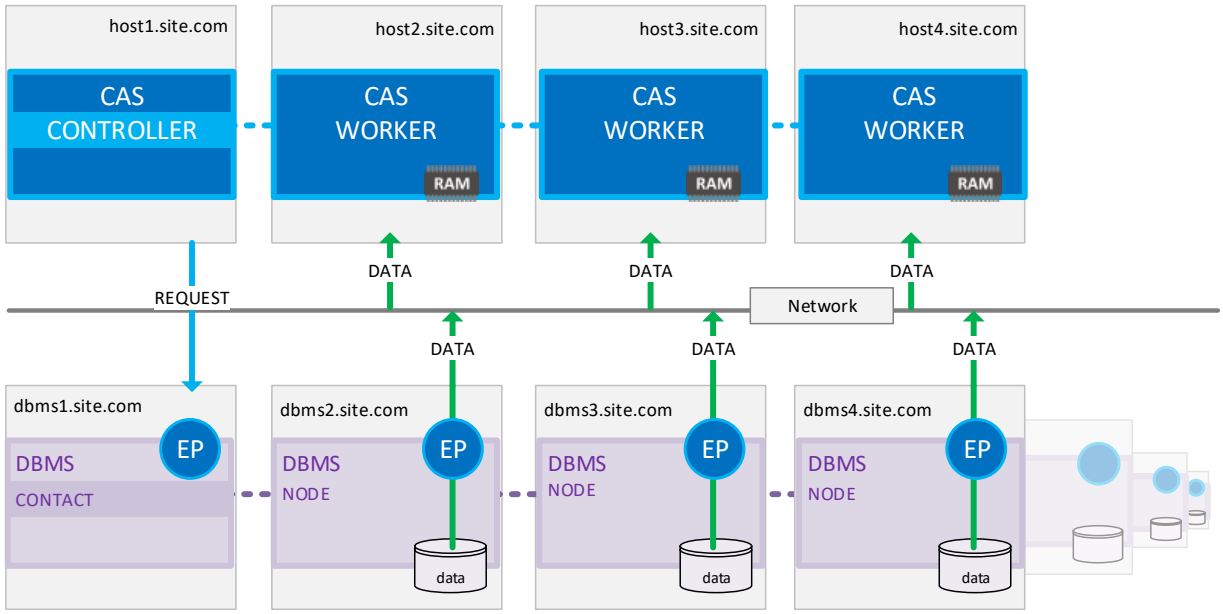

Figure 7. CAS Loads Data via the SAS In-Database Embedded Process from a Remote DBMS

In this illustration, we're showing the EP that is deployed to a remote DBMS. This allows parallel data loading to occur directly so that the EP on each DBMS host distributes its subset data to all CAS workers.

Table 2. High Performance Data Transfer Attributes of CAS with the SAS Embedded Process

Coordination	CAS controller directs workers for writes Each EP distributes to all workers for reads
Caslib srctype	HADOOP, SPDE, TERADATA
Format	Native RDBMS, SPD Engine files in HDFS, CSV files
Storage	Native RDBMS
Additional software required	SAS compatible DBMS SAS/ACCESS Interface to (DBMS) SAS In-Database Embedded Process SAS Data Connect Accelerator for (DBMS)

№ 3: SASHDAT ON DNFS

We're no longer limited to enjoying the benefits of the SASHDAT storage format on HDFS. SAS Viya provides a new parallel loading technology referred to as DNFS. This is a SAS specific acronym that references the generalized concept of a "distributed network file system." To be clear, this case is not limited just to standard NFS style solutions. The idea is that all CAS workers will be configured to use some form of centralized storage mounted on each CAS host at the same directory path to get to data. It's up to you (or your customer) if this means using network file system (NFS), network-attached storage (NAS), direct-attached storage (DAS), storage array network (SAN), redundant array of independent disks (RAID), just a bunch of disks (JBOD), and so on. The key is that all CAS hosts find data at the exact same local path.

In Figure 8, CAS accesses SASHDAT data in a SASDNFS container using the DNFS caslib.

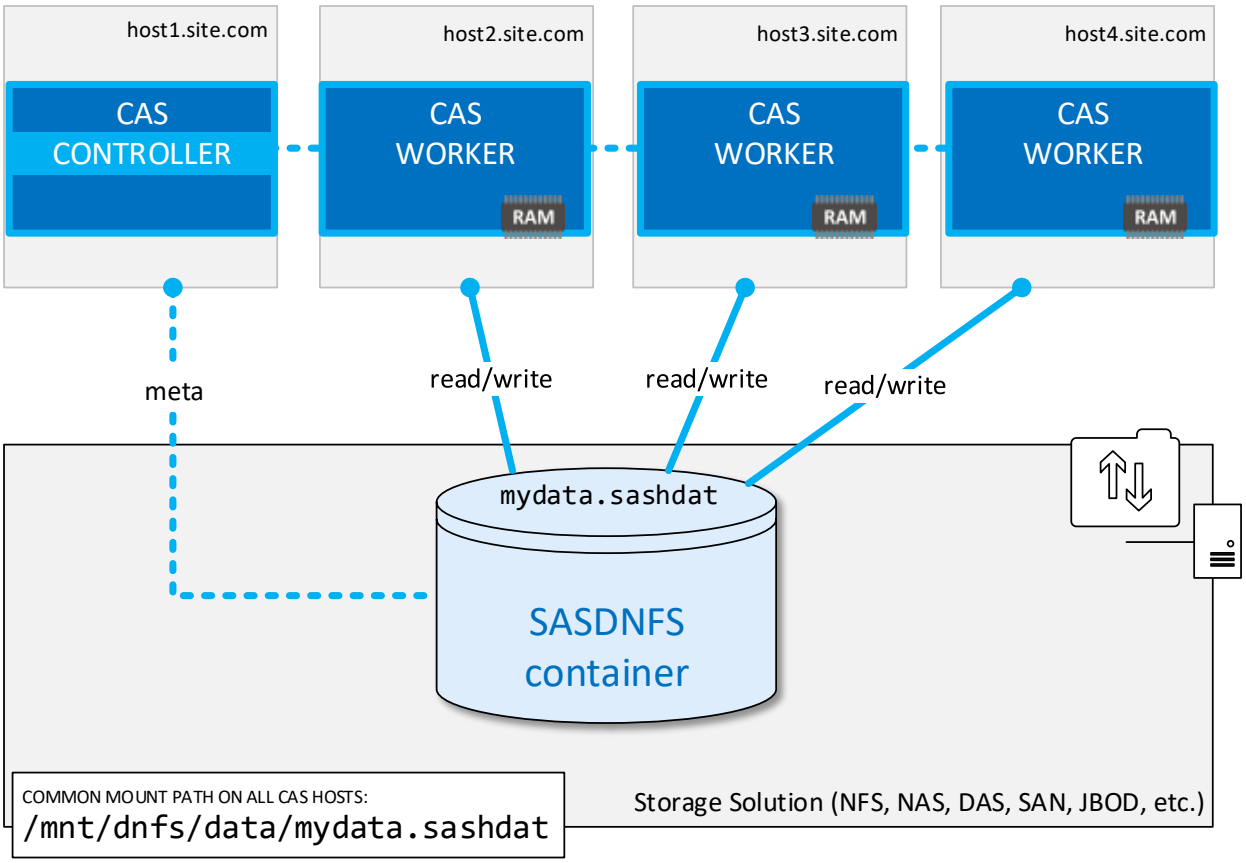

Figure 8. CAS Loads Data from SASHDAT via DNFS

Data stored using DNFS is also kept as blocks of SASHDAT, but these blocks are embedded all together inside a single file in a container format called SASDNFS. Even though the container is SASDNFS, the file type you'll see saved to disk is still ".sashdat".

The really cool thing about the SASDNFS container format is that it also works with the PATH type of caslib. When accessing SASDNFS files, the PATH type of caslib is for serial loading only – either for an SMP CAS Server or by the MPP CAS controller (for subsequent distribution to each of the CAS workers). This means that SASDNFS files can be copied using basic operating system commands – like cp, scp, rsync, and so on – to move or duplicate the data on disk between hosts of SMP and MPP CAS servers.

Table 3. High Performance Data Transfer Attributes of CAS using DNFS

Coordination	CAS controller directs workers
Caslib srctype	DNFS (for MPP CAS only, parallel transfer) PATH (for SMP or MPP CAS, serial transfer only)
Format	SASHDAT (in SASDNFS container), CSV files
Storage	Any POSIX compliant filesystem mounted to identical directory paths on all hosts of a CAS cluster. (Different CAS deployments can have different paths).
Additional software required	None

№ 4: BASE SAS ENGINE DATA SETS

For the first time, we now can read and write SAS data sets (SAS7BDAT files) in parallel. Like the infrastructure for DNFS, we need to ensure that all CAS hosts can directly access the SAS data sets:

- Place the SAS data set on a shared filesystem mounted on all CAS workers;
- Ensure that all CAS workers have direct access to the SAS7BDAT file at the same physical path;
- Specify a caslib datasource with srctype="PATH" and dataTransferMode="PARALLEL" or "AUTO".

Figure 9 illustrates the concept that even SAS data sets can be loaded into CAS using parallel I/O channels.

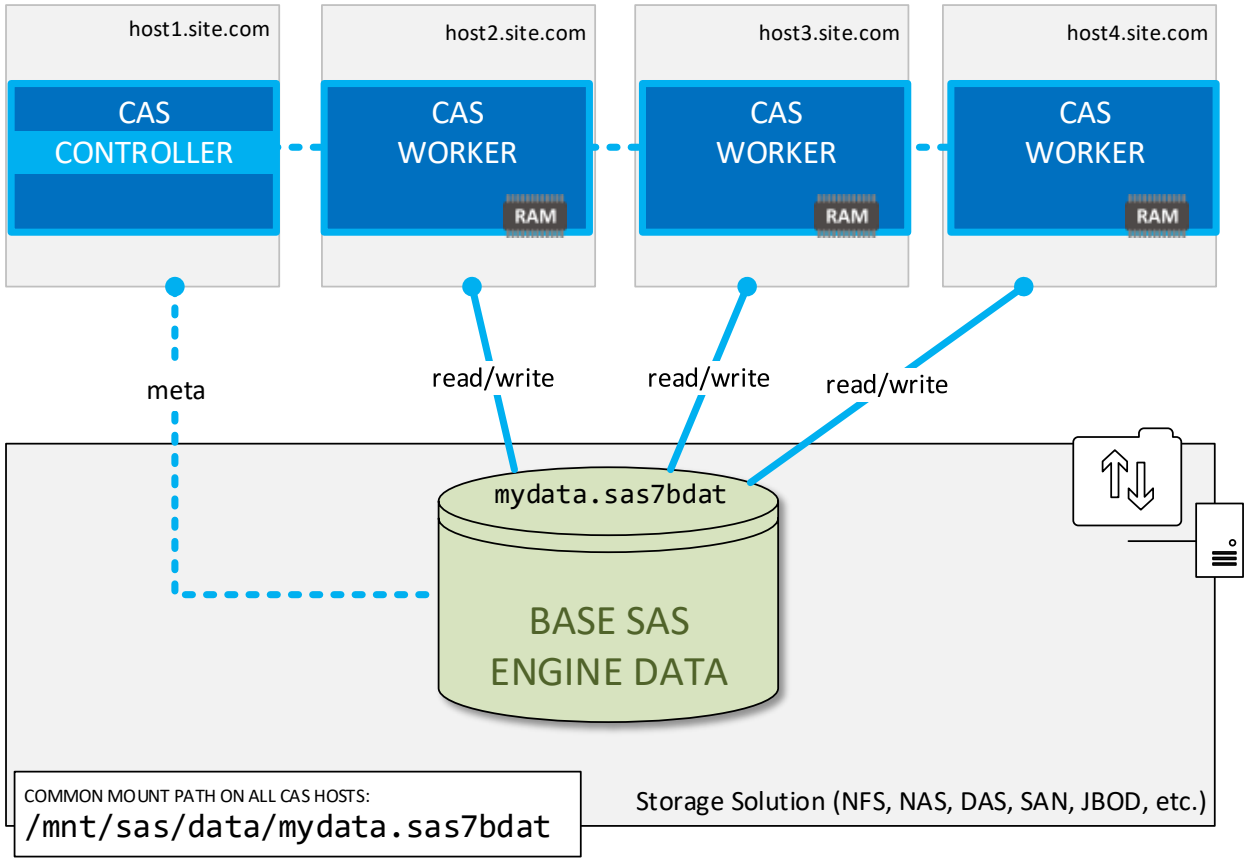

Figure 9. CAS Loads Data in Parallel from SAS7BDAT

If only the CAS controller has direct access to the SAS data set and if AUTO is the value for the DATATRANSFERMODE parameter, then the CAS controller can fall back to performing a serial transfer if the CAS workers don't have the necessary direct access. Otherwise, you can use the more conventional route of using SAS to read the data set and send it over the network to the CAS controller.

Table 4. High Performance Data Transfer Attributes of CAS with SAS Data Sets

Coordination	CAS controller directs workers
Caslib srctype	PATH

Format	SAS Data Sets (.sas7bdat)
Storage	Any POSIX compliant filesystem mounted to identical directory paths on all hosts of a CAS cluster. (Different CAS deployments can have different paths).
Additional parameters	Specify caslib attribute DATATRANSFERMODE=<"AUTO" or "PARALLEL">

№ 5: MULTI-NODE DATA TRANSFER

SAS Viya 3.3 introduces another new feature with our SAS/ACCESS and SAS Data Connector products called multi-node data transfer. The idea with multi-node data transfer is that the CAS controller can direct each participating CAS worker to request its own subset of rows from a table in a remote data provider. Ideally, this means that data is effectively loaded across multiple concurrent channels into CAS (that is, in parallel). But not all sites are ideal. There are often situations where there is a single point bottleneck, so CAS is very flexible and can accommodate a range of topologies.

In **Error! Reference source not found.**, CAS workers are shown performing multi-node data transfer where each worker makes its own query for its assigned subset of data, as coordinated by the CAS controller.

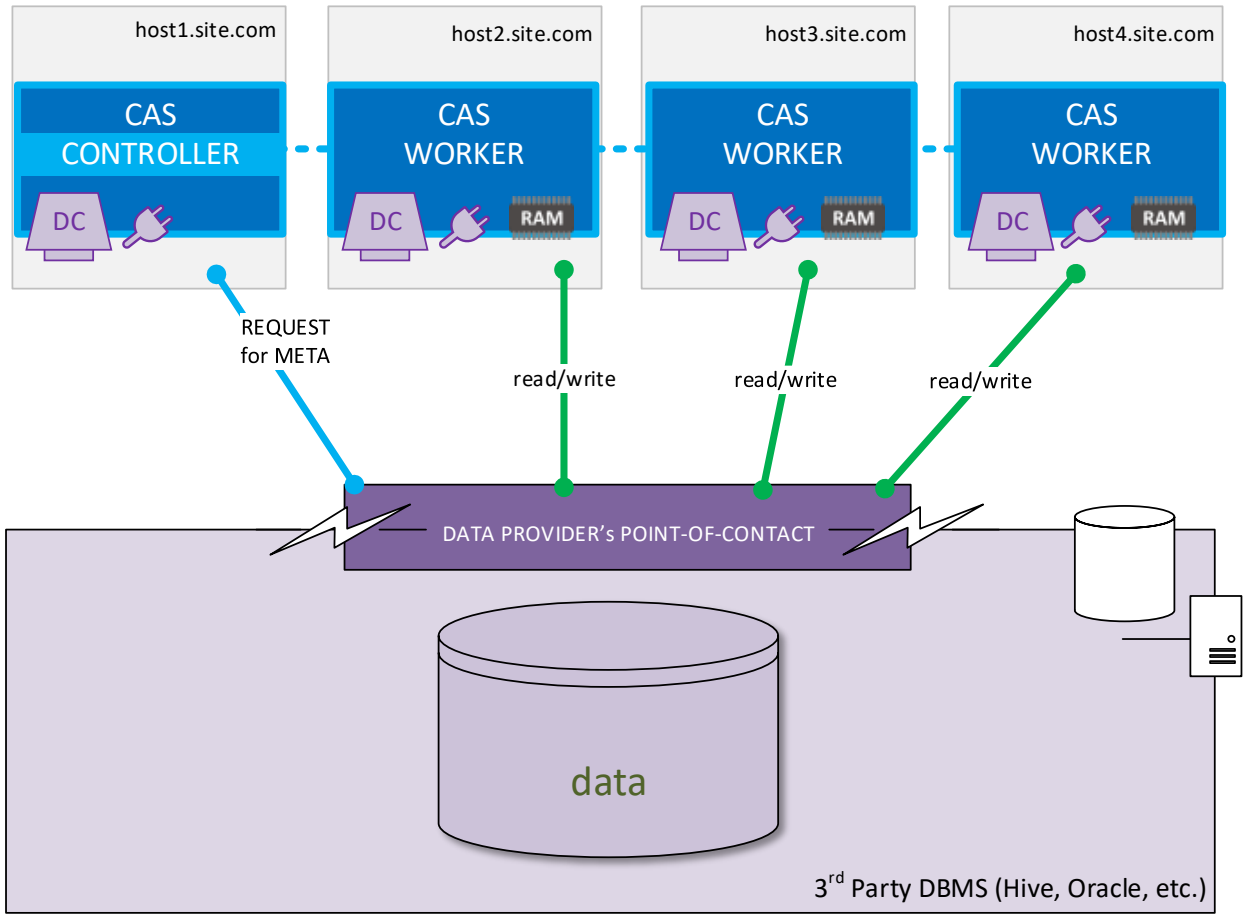

Figure 10. CAS Performs Multi-node Data Transfer from DBMS

Each data provider has its own communication procedures. The point-of-contact portion of the preceding figure is meant to accommodate the various service APIs that are implemented by different DBMS vendors.

To enable multi-node data transfer, ensure that your system has these features:

- CAS running in MPP mode
- A caslib defined to the remote data provider with DATATRANSFERMODE="SERIAL" (yes, serial)
- SAS/ACCESS and the associated SAS Data Connector are in place for the remote data provider (Note that multi-node data transfer isn't available for ODBC or PC file formats.)
- The official client for the remote data provider is installed and configured on the CAS hosts
- The remote data provider supports multiple concurrent connections
- The source table contains a numeric variable of type INT, DECIMAL, NUMERIC, or DOUBLE

Beyond those items, there are two more parameters in PROC CASUTIL we can set that will then direct CAS to attempt multi-node data transfer.

PROC CASUTIL:

NUMREADNODES= (for LOAD operations)

NUMWRITENODES= (for SAVE operations)

- "1" – specifies **serial** transfer where the CAS controller reads all data from source and distributes it evenly across the CAS workers
- "0" – specifies **multi-node** data transfer where the CAS controller directs all workers to participate, if possible.
- ">1" – specifies **multi-node** data transfer where the CAS controller directs the specified number of workers to participate, if possible

When CAS is directed to attempt a multi-node transfer of a table, the CAS controller contacts the data provider for information about the target table. Specifically, it's looking for an appropriate numeric-type of field as a means to split the data across participating CAS workers. After the controller gives the workers their assignments, each worker connects to the data provider and queries for its allotted data.

There is a lot of variation to be found from one customer site to another, even within implementations of a single vendor's DBMS product. So, when attempting multi-node data transfer, CAS proceeds with the directives that are given, but then it tests the environment to confirm that all is as expected. If there's an attribute of the environment that prevents CAS from performing multi-node exactly as directed, it attempts to gracefully fall back to the next best option. Here are some examples:

- To perform multi-node data transfer, CAS requires the target table in the data provider to contain an accepted numeric variable. If no suitable numeric variable is found, SAS proceeds with standard serial data transfer.
- For a CAS worker to participate in multi-node data transfer, it must have the associated data provider's client software installed locally and configured. If not, then the CAS controller excludes that worker from participating in the multi-node transfer.
- Some data providers limit the number of concurrent connections. So, for example, if the data provider only accepts five concurrent connections and we have directed PROC CASUTIL to use more than five CAS workers, then the CAS controller directs only five of its workers to make the transfer.

One important caveat to keep in mind is that it's the CAS client's responsibility to ensure that data is loaded as desired. So, if the CAS controller directs a subset its workers to perform a multi-node load, then those participating are the only workers with the data in memory. If your intention is for the data to be

distributed across all of your CAS workers, then that's an extra step. CAS does not automatically do it for you.

To redistribute data so that it's spread evenly across all of the CAS workers in your CAS session, use PROC CAS to submit the table.partition action:

```
/* Direct CAS to balance data evenly across all CAS workers */
PROC CAS;
  table.partition / table={caslib="cas_data",
                           name="cas_table",
                           groupByMode="REDISTRIBUTE"
                          };
RUN;
QUIT;
```

Table 5. High Performance Data Transfer Attributes of CAS Using Multi-Node

Coordination	CAS controller directs workers
Caslib srctype	REDSHIFT, DB2, HADOOP, IMPALA, SQLSERVER, ORACLE, POSTGRES, HANA, SPDE, TERADATA Multi-node isn't available for srctypes ODBC or PATH (PC files)
Format	Native DBMS
Storage	Native DBMS
Additional software required	SAS approved DBMS and DBMS client SAS/ACCESS Interface to (DBMS) SAS Data Connector to (DBMS)
Additional parameters	Specify caslib attribute DATATRANSFERMODE="SERIAL" Specify PROC CASUTIL attribute numREADnodes or numWRITEnodes with a value other than "1"

ADDITIONAL CONSIDERATIONS

There are a few more things it's helpful to keep in mind when bringing data over into CAS.

CAS DISK CACHE

CAS has the ability to protect against the unexpected loss of a worker. One mechanism it uses for this purpose is the CAS Disk Cache. This cache is dedicated space on disk which CAS can use as a temporary backing store for data in-memory. If a CAS worker unexpected goes down, then the other workers can recover its lost allocation of data from the block copies in their own cache on disk.

CAS, however, is selective about when it will use its disk cache. Generally speaking, CAS wants to memory-map to SASHDAT files on disk which represent the data it has in memory. If data comes from a non-SASHDAT source (like a remote DBMS), then CAS will load the data into memory and also place copies of that data in CAS Disk Cache as SASHDAT blocks.

On the other hand, if the source is already SASHDAT on direct-attached disk (like DNFS or symmetrically co-located HDFS), then CAS will memory-map to those blocks directly – and not copy to its cache. If a table in memory is modified, then change blocks will be placed in CAS cache to represent the difference.

One notable exception as of Viya 3.3 is when the CAS deployment is remote to (or placed asymmetrically alongside) HDFS. When loading, the blocks of data for the table must be copied over the network from the HDFS nodes to the CAS workers. In this situation, CAS does not automatically replicate in-memory blocks like we might expect. This means that there is no failover protection by default – it must be programmatically enabled by the client.

COMPATABILITY WITH SASHDAT FROM LASR

CAS can read legacy SASHDAT tables created by LASR stored in HDFS – assuming that all column sizes are compatible for transcoding to UTF-8. If they're not, then the table must be saved to the new CAS SASHDAT file format that increases the columns' variable length and transcodes to UTF-8 so that the CAS actions can then work with the data.

LASR cannot work with new SASHDAT files generated by CAS.

SERIAL ISN'T ALWAYS SERIAL

The DATATRANSFERMODE option for non-PATH type caslibs takes three values – and these may be the cause for confusion in determining if data is actually transferred in serial or in parallel to CAS. Those values are translated by CAS to drive its use of specific software products. Whether those products can deliver on the plain English meaning of the specified value is determined later.

Table 6. DATATRANSFERMODE Determines SAS Product to Use

DATATRANSFERMODE	CAS will use	Actual transfer method
PARALLEL	SAS Data Connect Accelerator	Only parallel
SERIAL	SAS Data Connector	Serial or multi-node if indicated. Automatically detect and gracefully fall back: Multi-node > Serial
AUTO	SAS Data Connect Accelerator. If fails, use SAS Data Connector.	Parallel if possible. Automatically detect and gracefully fall back: Parallel > Multi-node > Serial

When PARALLEL is specified as the DATATRANSFERMODE for caslibs other than the Path type, that explicitly locks CAS to working only with the SAS Data Connect Accelerator and the SAS Embedded Process. If the EP cannot perform multiple channel I/O to the CAS Workers, then the transfer attempt is stopped with an error in the SAS log.

As you can see, in SAS' technical terminology, multi-node data transfer falls under the SERIAL label. However, the ideal multi-node configuration will indeed allow for multiple, parallel I/O channels for transferring data from the data provider directly to each of the CAS Workers. Real-life implementations, as we all know, are not always ideal. When the NUMREADNODES (or NUMWRITENODES) attributes indicate multi-node transfer, CAS is programmed to verify multi-node capabilities and, if needed, it will reduce the number participating Workers to proceed with the transfer.

Use AUTO as the DATATRANSFERMODE for the most flexibility. CAS will attempt to "do the right thing" by gracefully degrading from parallel to multi-node all the way down to serial if necessary.

CONCLUSION

As data volumes continue to increase, the challenge to analyze it all quickly and efficiently is always present. Before that data can be analyzed in SAS Viya, it must be transferred from source systems into the memory space where CAS can work with it. Speeding up the time of this transfer helps realize the investment in massively parallel processing environments. SAS Viya and CAS offer new, unprecedented techniques to help achieve new options in pursuing faster data movement, analysis, and reporting.

RECOMMENDED READING

- *SAS® Viya® 3.3 for Linux: Deployment Guide*
- *SAS® Viya® 3.3: Data Preparation*
- *SAS® Viya® 3.3 Administration*
- *SAS® Cloud Analytic Services 3.3: User's Guide*
- *SAS/ACCESS 9.4 for Relational Databases: Reference*
- *SAS Viya Support for Databases*
  https://support.sas.com/en/documentation/third-party-software-reference/viya/support-for-databases.html

CONTACT INFORMATION

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

Rob Collum
Principal Technical Architect, Scalability Lead
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
rob.collum@sas.com
http://www.sas.com/