Paper SAS3479-2019

Seriously Serial or Perfectly Parallel Data Transfer with SAS® Viya®

Rob Collum, SAS Institute Inc.

ABSTRACT

SAS® Cloud Analytic Services (CAS) is the engine of our in-memory analytics offerings. It promises to deliver massively scalable performance of huge volumes of data. But then getting all that data loaded into CAS (or saved back out later) is another scalability challenge to tackle. CAS addresses that challenge by providing support for a wide range of data sources starting with native SAS® formats and extending to include direct loading from many third-party data providers. Along the way, we also need to carefully plan and evaluate the architecture and infrastructure to ensure that everything is moving along as fast and efficiently as intended.

As of SAS® Viya® 3.4, the language we use to direct CAS in its efforts to transfer data has evolved considerably. We can explicitly configure CAS to use a serial data transfer method, but the objective we're shooting for is effectively parallel. Alternatively, we can set up CAS and other related components for a fully parallel transfer, but then CAS might deem it necessary to downgrade to a serial approach.

So let's crack open this case and look at determining exactly what type of data transfer we're asking CAS to perform, verifying that's what it did, and understanding why it worked.

INTRODUCTION

The infrastructure to run large-scale SAS Viya solutions often relies on significant investment in infrastructure, personnel, and ongoing operations. Getting the most value out of that investment is important. One aspect to consider is how large volumes of data are transferred into and out of the SAS Cloud Analytic Server.

Serial data transfer occurs when data is moved from beginning to end as a sequential stream of bits. This conventional technique is the most common and flexible way to move data around with SAS software.

Parallel data transfer is achieved when data is moved across multiple concurrent channels. This effectively multiplies the rate of data transfer, which allows SAS Cloud Analytic Services (CAS) to read and write very large volumes of data in a fraction of the time that serial movement would allow. CAS supports parallel data transfer with data hosted in select storage formats in combination with certain file hosting services.

CAS also supports multi-node data transfer. This is a concept where multiple connections are attempted in expectation of parallel transfer, but resilient to environmental limitations that might result in limited concurrency, even serial movement. For this paper, multi-node data transfer is grouped with the concept of parallel data movement in general.

A site's ability to perform serial or parallel data transfer requires an intimate understanding of the infrastructure, software licensing, third-party data source capabilities, and more. CAS will default to either serial or parallel data transfer as is appropriate for the specified data source. That default can be overridden in some cases with specific SAS programming syntax.

While there is no consistent, universal syntax in SAS programming to guarantee either serial or parallel data movement in all situations, this paper will guide the reader to achieve

the desired outcome. When writing SAS code to perform data transfer with CAS, careful consideration of options and defaults is required to ensure that data moves as expected.

SERIAL DATA TRANSFER

Serial data movement is simple to understand. It's the sequential movement of data from one location to another along a single pathway as shown in Figure 1.

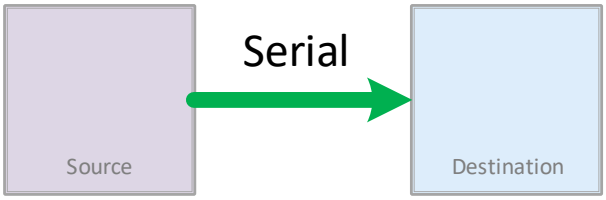

Figure 1. Serial is the Simple Movement of Data from Source to Destination Along a Single Path

Serial data transfer is always available to SAS Cloud Analytic Services. For an SMP deployment of CAS (that is, CAS deployed to run on a single host machine), then data movement will be serial by definition.

Where things get interesting is when working with an MPP (massively parallel processing) deployment of CAS where the CAS software is deployed to run as a single logical server across multiple host machines.

Serial transfer for MPP CAS has a unique meaning. Figure 2 shows the CAS controller reading the data from source and round-robin distributing that data evenly (or as partitioned) to the CAS workers.

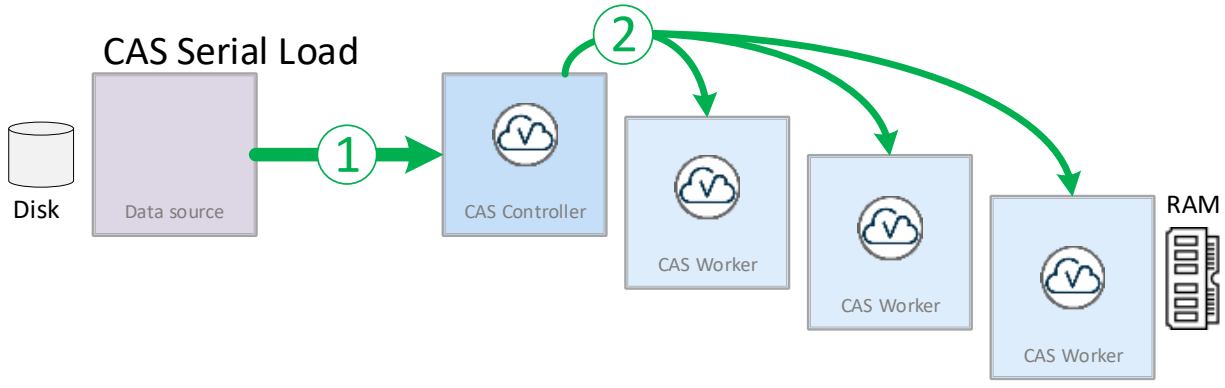

Figure 2. Data Loading Serially with MPP CAS

Serial data transfer is the most flexible approach for loading data into CAS. That is, every source of data which CAS has the ability to communicate with can perform serial transfer to load data into CAS (or save it back) – with one exception, SASHDAT on co-located HDFS which is parallel only.

PARALLEL DATA TRANSFER

Parallel data transfer is intended to multiply the rate at which data can be moved. Figure 3 illustrates the idea that multiple, independent paths are used concurrently to move data from source to destination.

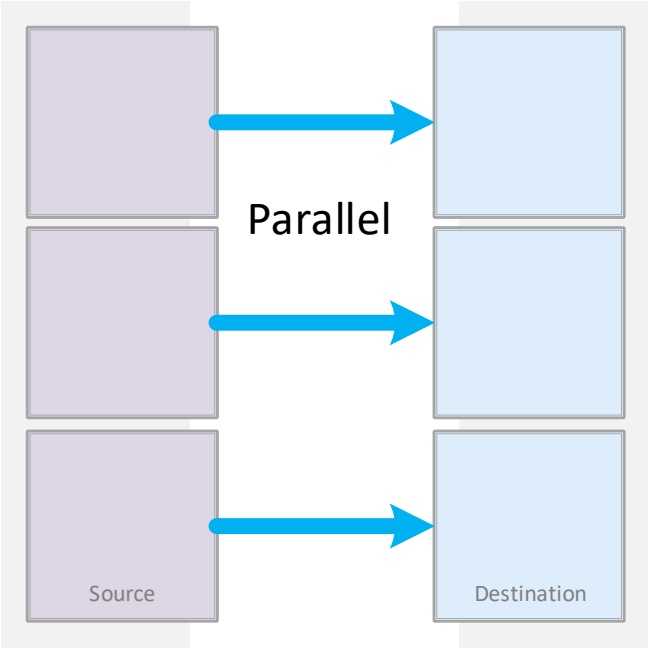

Figure 3. Parallel Multiplies the Rate of Data Transfer

MPP CAS is the required topology for parallel data transfer. The CAS controller coordinates parallel data movement and directs the CAS workers' actions depending on the circumstance. Those circumstances dictate which the way in which parallel data movement can occur. The kind of parallel data movement depends on the data source, the data format, the software in use, and the physical infrastructure.

The first form of parallel transfer of data to a SAS in-memory analytics engine was offered with the SAS LASR Analytic Server (before Viya and CAS were introduced). Data was placed in the proprietary SASHDAT format and stored in the Hadoop Distributed File System (HDFS) which was installed symmetrically alongside LASR on the same host machines. This technique provides the fastest, most efficient technique for (re-)loading data into LASR. Now with SAS Viya, CAS continues to offer this same access as shown in Figure 4.

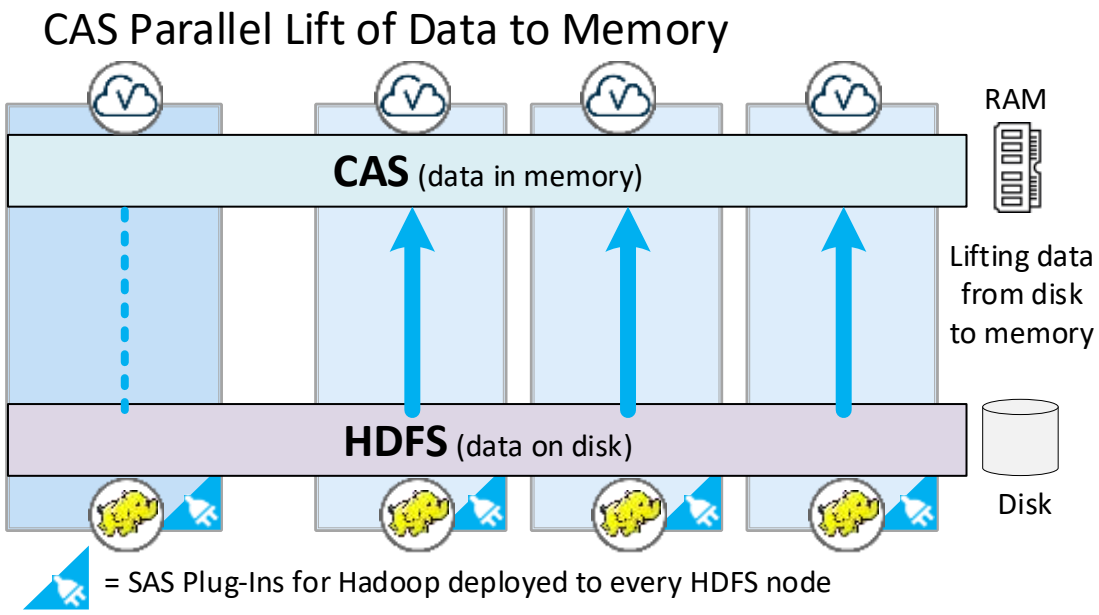

Figure 4. Working in Parallel, Each CAS Worker Reads SASHDAT Data from Its Local HDFS Data Node, Lifting That Data to Memory on the Same Host

While Hadoop has proven itself as a cost-effective means of storing data using a shared-nothing concept, there are many sites which have not made the investment in Hadoop technology or expertise. To address the needs of those customers, CAS also offers the ability to read SASHDAT data in parallel from a shared file system – a technology SAS refers to as "DNFS" – as shown in Figure 5.

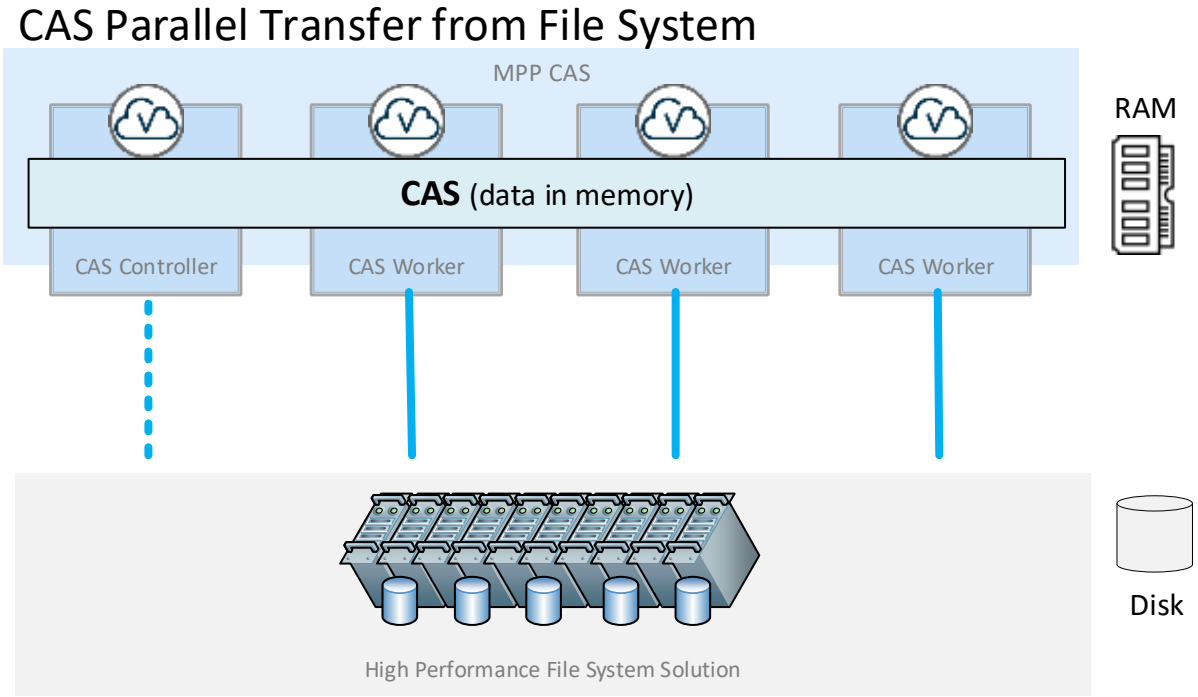

Figure 5. CAS Workers Reading Data in Parallel from a Shared File System

Another parallel approach which CAS has inherited from its LASR predecessor is the ability to transfer data directly from the SAS In-Database Embedded Process (EP). The key tenet of the EP is to take SAS analytic power to where the data already resides. Beyond that effort, data may then need to move to CAS for further advanced analysis, business reporting, and more. When the EP is deployed to a data provider on hosts separate from CAS, then each instance of the EP will round-robin distribute its allotment of the data evenly (or as partitioned) to all of the CAS workers. Because each CAS worker handles multiple concurrent connections, we sometimes refer to this as a massively parallel transfer as shown in Figure 6.

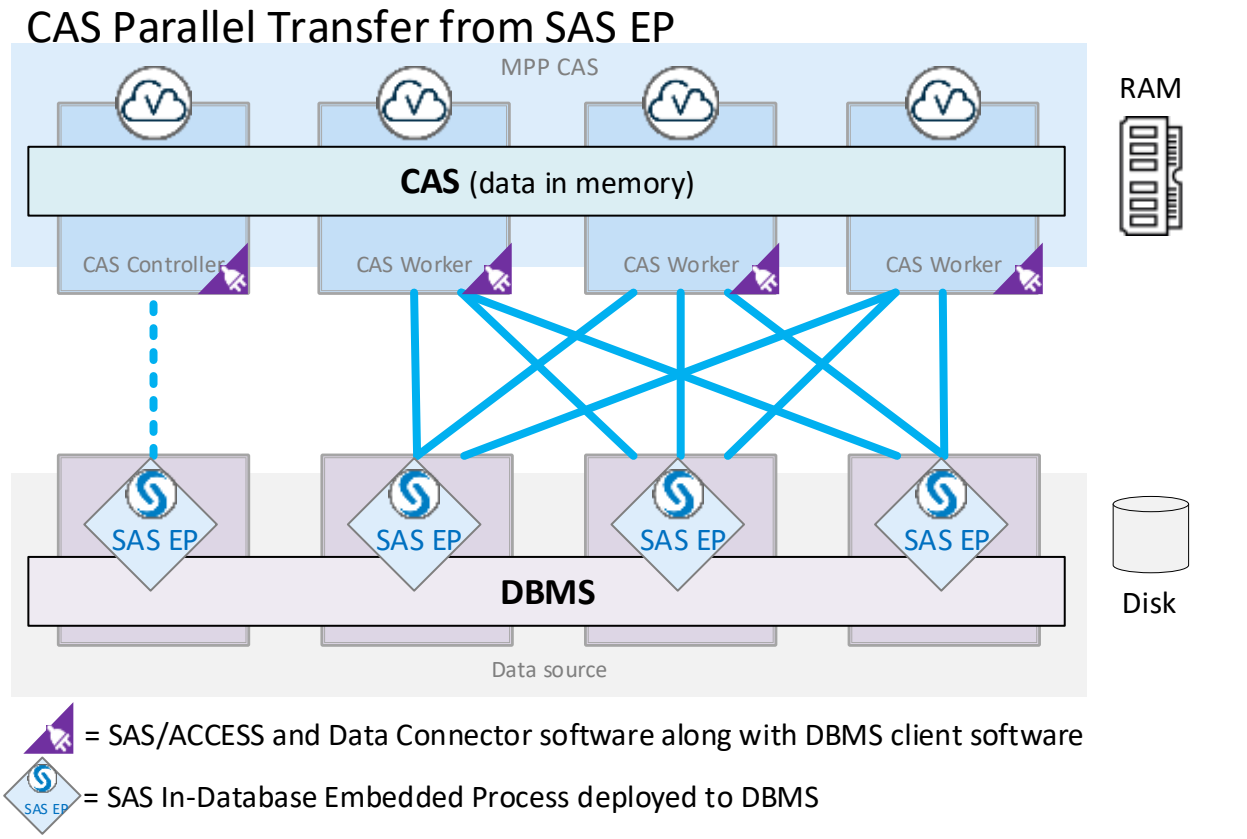

Figure 6. Each CAS Worker Uses Multiple Concurrent Connections to Transfer Data from the SAS Embedded Process

There's another form of (ideally) parallel movement which is exclusive to CAS: Multi-node transfer. Multi-node transfer is a feature of SAS Data Connectors offered with SAS/ACCESS products. The term multi-node was chosen to reflect the reality that while CAS may attempt multiple parallel channels of data transfer, the effective data movement may be serial if relying on a single point of contact. Figure 7 shows one possible scenario of many for multi-node transfer.

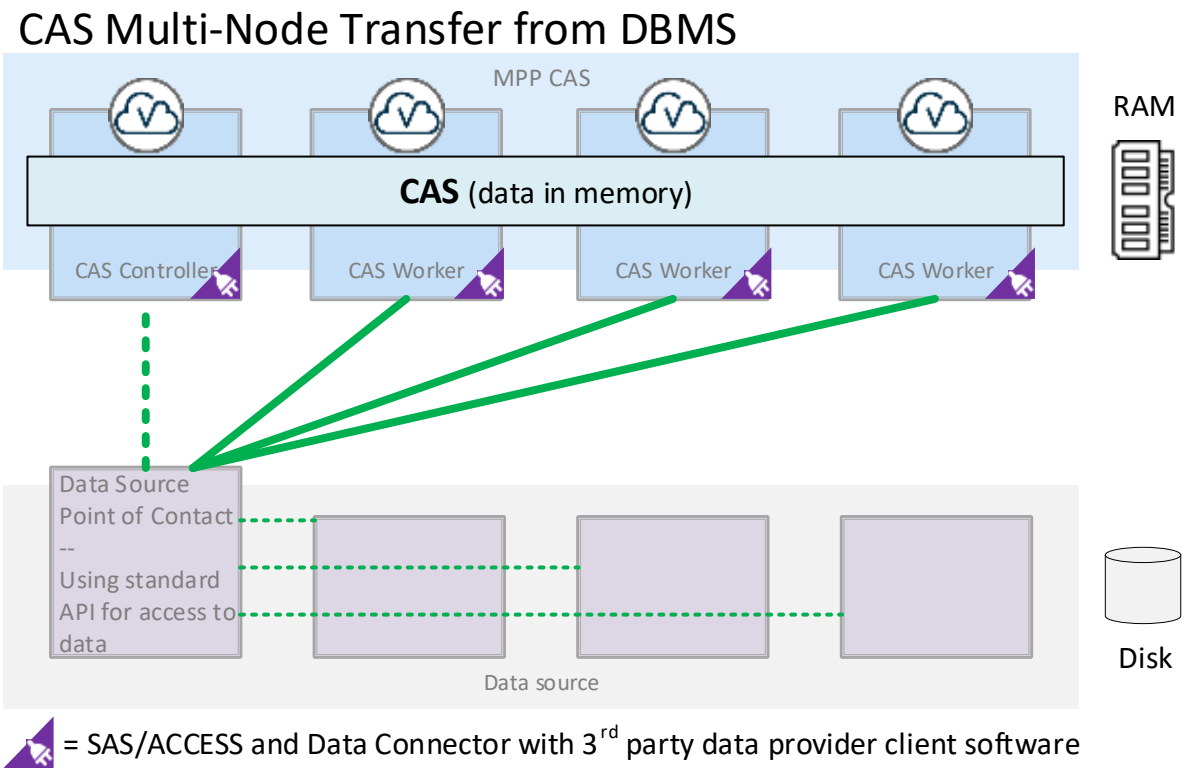

Figure 7. Multi-Node Transfer is a Feature of SAS/ACCESS and Data Connectors with CAS

Multi-node transfer has some unique characteristics:

- Multi-node is offered by SAS Data Connectors provided by SAS/ACCESS products. Its functionality is not offered for all parallel data sources or file types.
- Multi-node is highly dependent on the data source and intermediate architecture. Even if multiple CAS workers participate simultaneously, the transfer of data may still be effectively serial if that's what the environment provides.
- In older versions of SAS Viya, the SAS Data Connectors could only perform serial transfer. Hence, in SAS syntax (and log messages), load (or save) actions described as "serial" typically mean that the SAS Data Connector was used. And so, in SAS programming lingo, multi-node transfer is also "serial" even if we know it's effectively parallel.
- Multi-node requires the source data to have a suitable numeric variable on which to perform splits – not just any numeric will work.
- Multi-node can always fallback to use fewer CAS workers if the situation calls for it. Fallback is automatic.
- Data transferred using multi-node is kept on the CAS workers which participated in the transfer. If you desire the data to be available to all the CAS workers, then that re-distribution is a separate step.

The effectiveness and availability of parallel data transfer assumes that CAS is equipped with the software components required, that the infrastructure is sufficient to accommodate multiple concurrent I/O channels, and the format, structure, and content of the data itself.

In some cases, each CAS worker might establish multiple connections to the data source to perform the transfer. This can further enhance and/or complicate data movement as available network capacity is consumed for the transfer.

NATIVE DATA SOURCES

SAS Cloud Analytic Server can work with several data formats and providers natively – that is, no additional SAS/ACCESS software is necessary. For those sources, CAS still relies on SAS Data Connector *technology* as an extensible model for data access.

►The PATH Type of CASLIB – serial, except when parallel

A caslib with a srcType value of PATH is typically used to reference data on disk which is locally accessible to the CAS controller. Since the controller usually does the work of loading (or saving) the data alone, data transfer is most often, but not always, serial.

Serial Transfer with PATH Type of CASLIB

For file types of CSV, SASHDAT, and XLS/XLSX (Microsoft Excel spreadsheets and workbooks), CAS will only perform serial load and save of data.

The following code shows sample syntax for serial data transfer using the PATH type of CASLIB:

```
caslib mypath sessref=mysess1 path="/path/to/data"
             datasource=(srcType='path');

proc casutil;
   load casdata="carousel.csv" casout="carousel_csv"
   importoptions=(filetype="csv") ;
run;
```

Run the code above to get the following SAS log results:

```
77    caslib mypath sessref=mysess1 path="/path/to/data"
78                  datasource=(srcType='path');
NOTE: 'MYPATH' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYPATH'.
NOTE: Action to ADD caslib MYPATH completed for session MYSESS1.
79    proc casutil;
NOTE: The UUID '13cc4311-baf1-c248-be08-23e4a89308ae' is connected using
session MYSESS1.
80       load casdata="carousel.csv" casout="carousel_csv"
81       importoptions=(filetype="csv") ;
```

```
NOTE: Cloud Analytic Services made the file carousel.csv available as table
CAROUSEL_CSV in caslib MYPATH.

NOTE: The Cloud Analytic Services server processed the request in 0.612032
seconds.

82    run;
```

When serial-loading is the only possible option for CSV, SASHDAT, and XLS/XLSX file formats, the SAS log does not indicate the actual dataTransferMode used to load that data into CAS.

Parallel Transfer of SAS7BDAT with PATH Type of CASLIB

The one notable exception to the expected data transfer of PATH type CASLIBs is standard SAS data sets (SAS7BDAT). When the filetype importOption is BASESAS (either specified or automatically determined), CAS will default to a dataTransferMode of AUTO. This means that CAS will attempt a parallel transfer using all session workers if possible. If parallel transfer fails for some reason, then it will try serial transfer.

The following code shows sample syntax for transferring SAS7BDAT files using the PATH type of CASLIB:

```
caslib mypath path="/path/to/data" sessref=mysess1
             datasource=(srcType='path');

proc casutil;
   load casdata="baseball.sas7bdat" casout="baseball_bdat"
   importoptions=(filetype="basesas", dtm="auto", debug="dmsglvli") ;
run;
```

Run the code above to get the following SAS log results:

```
76    caslib mypath path="/path/to/data" sessref=mysess1
77                   datasource=(srcType='path');
NOTE: 'MYPATH' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYPATH'.
NOTE: Action to ADD caslib MYPATH completed for session MYSESS1.
78
79    proc casutil;
NOTE: The UUID '1581d4f8-e134-5c4e-9991-b0dc79350f19' is connected using
session MYSESS1.
80       load casdata="baseball.sas7bdat" casout="baseball_bdat"
81       importoptions=(filetype="basesas" dtm="auto" debug="dmsglvli") ;
NOTE: BASE Data Connector, Using PARALLEL mode
NOTE: Cloud Analytic Services made the file baseball.sas7bdat available as
table BASEBALL_BDAT in caslib MYPATH.
```

```
NOTE: The Cloud Analytic Services server processed the request in 0.086559
seconds.

82    run;
```

Notice also that a new directive was provided in the importOptions: debug="dmsglvli". When present, CAS will log how the data was transferred (serial or parallel). This debug option is only available for use when the fileType="basesas" option is also specified.

Only loading actions for SAS7BDAT files into CAS can occur in parallel. Saving SAS7BDAT back to the PATH CASLIB is always serial.

PATH CASLIB Filetypes	PROC importOptions	Data Transfers	Log Shows Actual Transfer Mode
SASHDAT	None	Serial only	No
SAS7BDAT (loading)	importOptions=(fileType="basesas" dataTransferMode='AUTO')	Parallel if possible, Else serial	Yes, with debug="dmsglvli"
	importOptions=(fileType="basesas" dataTransferMode='PARALLEL')	Parallel only	Yes, with debug="dmsglvli"
	importOptions=(fileType="basesas" dataTransferMode='SERIAL')	Serial only	Yes, with debug="dmsglvli"
SAS7BDAT (saving)	None	Serial only	Yes, with debug="dmsglvli"
CSV, XLS, XLSX	None	Serial only	No
	None	Serial only	No

Table 1. Data Transfer Modes for Filetypes in a PATH Type of CASLIB

Minimizing Confusion

Key takeaways from data transfer using the PATH type of CASLIB are as follows:

- Options shown in purple in Table 1 (and other tables below) are the defaults used unless otherwise specified.
- Caslibs with a srcType of Path usually use serial transfer to load files (and always serial to save files). The exception is when loading SAS data sets (SAS7BDAT files) which default to parallel transfer.

- The SAS log does not indicate whether serial or parallel transfer is used. The exception is when loading SAS data sets (SAS7BDAT files) and specifying the importOption debug="dmsglvli".

►The DNFS Type of CASLIB – expected to be parallel

The DNFS type of CASLIB is a new parallel data transfer technology introduced with CAS for accessing data using conventional disk storage solutions. The idea with DNFS is that a shared filesystem is mounted to identical directory paths on the CAS controller and all CAS workers. The CAS controller can then coordinate and direct the CAS workers to grab their specific allotment of rows from disk.

The DNFS type of CASLIB is typically used to work with the SASHDAT file format, which is optimized for parallel data movement. The DNFS CASLIB can also transfer data using the CSV text file format.

The following code shows sample syntax for transferring data using the DNFS type of CASLIB:

```
caslib mydnfs path="/path/to/data"
             sessref=mysess1 datasource=(srcType='dnfs');

proc casutil;
   load casdata="domino.sashdat" casout="domino_hdat";
run;
```

Run the code above to get the following SAS log results:

```
77    caslib mydnfs path="/path/to/data" sessref=mysess1
78                   datasource=(srcType='dnfs');
NOTE: 'MYDNFS' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYDNFS'.
NOTE: Action to ADD caslib MYDNFS completed for session MYSESS1.
79
80    proc casutil;
NOTE: The UUID '1581d4f8-e134-5c4e-9991-b0dc79350f19' is connected using
session MYSESS1.
81       load casdata="domino.sashdat" casout="domino_hdat";
NOTE: Cloud Analytic Services made the DNFS file domino.sashdat available
as table DOMINO_HDAT in caslib MYDNFS.
NOTE: The Cloud Analytic Services server processed the request in 0.040622
seconds.
82    run;
```

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.

DNFS Filetypes	Data Transfers	Log Shows Actual Transfer Mode
SASHDAT	Parallel only	No
CSV	Parallel only	No

Table 2. Data Transfer Modes for Filetypes in a DNFS Type of CASLIB

Minimizing Confusion

Key takeaways from data transfer using the DNFS type of CASLIB are as follows:

- The DNFS type of CASLIB only performs parallel data transfer.
- The SAS log will not indicate parallel transfer.
- Refer to the Recommended Reading section below for the web article titled "SAS® Data Connector/SAS® Data Connect Accelerator Logging" for more information.

►The HDFS Type of CASLIB – pretty much always parallel

Like its predecessor (SAS® LASR™ Analytic Server), CAS can access data directly from the Hadoop Distributed File System (HDFS). With the HDFS type of CASLIB, the CAS controller coordinates and directs the CAS workers to load their specific allotment of blocks from HDFS.

The HDFS type of CASLIB is typically used to work with the SASHDAT file format, which is optimized for parallel data movement. The HDFS CASLIB can also transfer data using the CSV text file format.

The following code shows sample syntax for transferring data using the HDFS type of CASLIB:

```
caslib myhdfs sessref=mysess1 path="/path/to/data"
              datasource=(srcType='hdfs');

proc casutil;
   load casdata="honcho.sashdat" casout="honcho_hdat";
run;
```

Run the code above to get the following SAS log results:

```
76    caslib myhdfs sessref=mysess1 path="/hdfs/path/to/data"
77                  datasource=(srcType='hdfs');
```

```
NOTE: 'MYHDFS' is now the active caslib.

NOTE: Cloud Analytic Services added the caslib 'MYHDFS'.

NOTE: Action to ADD caslib MYHDFS completed for session MYSESS1.

78

79    proc casutil;

NOTE: The UUID 'd1d4d0a7-404b-1945-9292-17e60e31f91b' is connected using
session MYSESS1.

80       load casdata="honcho.sashdat" incaslib="myhdfs"
casout="honcho_hdat";

NOTE: Cloud Analytic Services made the HDFS file honcho.sashdat available
as table HONCHO_HDAT in caslib MYHDFS.

NOTE: The Cloud Analytic Services server processed the request in 7.47437
seconds.

81    run;
```

The resilient nature inherent to HDFS practically guarantees distribution of data blocks across multiple Hadoop data nodes. Therefore, CAS only expects to perform parallel data transfer. For that reason, the SAS log does not indicate the actual dataTransferMode used to load the data into CAS. The definition of HDFS means that data transfers in parallel such that the CAS workers participate in working with the data directly from source.

Like DNFS, there's no programming option to specify a serial-only transfer of data from HDFS. Due to the nature of HDFS, it's unlikely we'll hit a serial-only aspect of the infrastructure. Our biggest concern might be the potential for network or process bottleneck for some remote/asymmetric deployments. However, as a rule, we should always expect that multiple, parallel I/O channels are used when CAS is working with SASHDAT files in HDFS.

HDFS Filetypes	Data Transfers	Log Shows Actual Transfer Mode
SASHDAT	Parallel only	No
CSV	Parallel only	No

Table 3. Data Transfer Modes for Filetypes in an HDFS Type of CASLIB

Minimizing Confusion

Key takeaways from data transfer using the HDFS type of CASLIB are as follows:

- The HDFS type of CASLIB only performs parallel data transfer.
- The SAS log will not indicate parallel transfer.
- Refer to the Recommended Reading section below for the web article titled "SAS® Data Connector/SAS® Data Connect Accelerator Logging" for more information.

► The LASR Type of CASLIB – parallel if possible, else serial

CAS can read SASHDAT data directly from the SAS LASR Analytic Server (but not write back). To specify parallel or serial transfer, use the parallelMode importOption in the PROC statement.

The following code shows sample syntax for transferring data using the LASR type of CASLIB:

```
caslib mylasr sessref=mysess1
              datasource=(srcType='lasr'
                          server="lasr.site.com" port=10031
                          username="lasruser" password="lasrpass"
                          signer="http://
lasr.site.com:80/SASLASRAuthorization"
                          );

proc casutil;
    load casdata="LIBREF.lyrics" casout="lyrics_lasr"
    importoptions=(filetype="lasr"  parallelmode="fallback")
   ;
run;
```

Run the code above to get the following SAS log results:

```
77    caslib mylasr sessref=mysess1
78                  datasource=(srcType='lasr'
79                              server="lasr.site.com" port=10031
80                              username="sasdemo" password=XXXXXXXX
81
signer="http://host.site.com:80/SASLASRAuthorization"
82                              );
NOTE: 'MYLASR' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYLASR'.
NOTE: Action to ADD caslib MYLASR completed for session MYSESS1.
83
84    proc casutil;
NOTE: The UUID '6acc7081-c4f1-3248-8bab-a1d59e453c16' is connected using
session MYSESS1.
85        load casdata="LIBREF.lyrics " casout="lyrics_lasr"
86        importoptions=(filetype="lasr"  parallelmode="fallback");
NOTE: Cloud Analytic Services made the external data from LIBREF.LYRICS
available as table LYRICS_LASR in caslib MYLASR.
```

```
NOTE: The Cloud Analytic Services server processed the request in 8.024644
seconds.
87     run;
```

Assuming that both LASR and CAS are running as distributed deployments (MPP on multiple hosts), the CAS workers can communicate directly with the LASR data nodes to fetch data in parallel. The number of workers and data nodes does not need to match. If either CAS or LASR is running as a single SMP server, then parallel transfer is not possible. The default value for parallelMode is FALLBACK, which directs CAS to attempt direct, parallel connections between its workers and LASR's data nodes. If that fails, then CAS will try again as a serial transfer between the CAS controller and the LASR root node.

LASR Filetype	PROC importOptions	Data Transfers	Log Shows Actual Transfer Mode
SASHDAT	`importOptions=(fileType="lasr" parallelmode="FALLBACK")`	Parallel if possible, Else serial	No
	`importOptions=(fileType="lasr" parallelmode ="FORCE")`	Parallel only	No
	`importOptions=(fileType="lasr" parallelmode ="NONE")`	Serial only	No

Table 4. Data Transfer Modes for Filetypes in a LASR Type of CASLIB

The value of the parallelMode importOption is effectively ignored when MPP CAS is loading data from SMP LASR since only serial transfer is possible.

Minimizing Confusion

Key takeaways from data transfer using the LASR type of CASLIB are as follows:

- CASLIBs with a srcType of LASR default to loading data (SASHDAT only) in parallel from MPP LASR to MPP CAS.
- Specify the parallelMode importOption if needed. The exception is when loading data from SMP LASR, which is serial only and the value specified for parallelMode is ignored.
- The SAS log does not indicate whether serial or parallel transfer is used.
- Refer to the Recommended Reading section below for the web article titled "SAS® Data Connector/SAS® Data Connect Accelerator Logging" for more information.

► The (Amazon) S3 Type of CASLIB – parallel sometimes, serial others

CAS can read and write SASHDAT data in Amazon S3 storage service in parallel only. However, CSV files are accessed in serial only.

The following code shows sample syntax for transferring data using the Amazon S3 type of CASLIB:

```
caslib mys3 datasource=(srctype="s3",
                        accesskeyid="GoBblEdYg0oK",
                        secretaccesskey="sAsSaFrAsS/kErCh0w",
                        region="US_East",
                        bucket="mybucket",
                        objectpath="/myfolder/"
                        );

proc casutil incaslib="mys3" outcaslib="mys3";
   load casdata="sushi.csv" casout="sushi_s3";
quit;
```

Run the code above to get the following SAS log results:

```
78    caslib mys3 datasource=(srctype="s3",
79                            accesskeyid="GoBblEdYg0oK",
80                            secretaccesskey="sAsSaFrAsS/kErCh0w",
81                            region="US_East",
82                            bucket="mybucket",
83                            objectpath="/myfolder/"
84                            );
NOTE: 'MYS3' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYS3'.
NOTE: Action to ADD caslib MYS3 completed for session MYSESS1.
85
86    proc casutil incaslib="mys3" outcaslib="mys3";
NOTE: The UUID '3e2390ca-9b1f-f749-b907-d6bc2b6d43a4' is connected using
session MYSESS1.
87       load casdata="sushi.csv" casout="sushi_s3";
NOTE: Cloud Analytic Services made the file sushi.csv in AWS S3 bucket
mybucket available as table SUSHI_S3 in caslib mys3.
NOTE: The Cloud Analytic Services server processed the request in 4.600251
seconds.
87    quit;
```

The S3 type of CASLIB is new in SAS Viya 3.4. Even though CSV files are only accessed using serial transfer (that is, only the CAS controller accesses the data in S3), all the CAS workers also require network visibility to the Amazon S3 environment (that is, to the FQDN of Amazon S3 servers, including bucket hosts). Amazon S3 supports a maximum file size of 5 TB. Transport Layer Security (TLS) encryption can be specified to ensure secure data movement.

Filetype in S3	Data Transfers	Options
SASHDAT	Parallel only	None
CSV	Serial only	None

Table 5. Data Transfer Modes for Filetypes in an Amazon S3 Type of CASLIB

Minimizing Confusion

Key takeaways from data transfer using the Amazon S3 type of CASLIB are as follows:

- CAS will only perform parallel loading of SASHDAT files stored in Amazon S3. No fallback to serial transfer is offered.
- Only the CAS controller will access CSV files, which reside in Amazon S3. Ensure that the CAS workers have network visibility to the S3 host machines as well.
- The SAS log does not indicate whether serial or parallel transfer is used.
- Refer to the Recommended Reading section below for the web article titled "SAS® Data Connector/SAS® Data Connect Accelerator Logging" for more information.

► The SPDE Type of CASLIB – parallel, unless serial is specified

CAS can read data files in the SAS Scalable Performance Data Engine (SPD Engine) format. Those tables can be stored either on disk mounted locally to CAS or in HDFS. Multi-node transfer using the SAS Data Connector for the SPD Engine is the default.

The following code shows sample syntax for transferring data using the SPDE type of CASLIB:

```
caslib myspde dataSource=(srctype="spde",
                          dataTransferMode="serial",
                          numReadNodes=0,
                          mdfpath="/path/to/spde/meta",
                          datapath="/path/to/spde/data",
                          dfdebug=sqldetails) ;

proc casutil ;
   load  incaslib="myspde" casdata="sparkle"
        outcaslib="myspde" casout="sparkle_spde" replace ;
run ;
```

Run the code above to get the following SAS log results:

```
77    caslib myspde dataSource=(srctype="spde",
78                              dataTransferMode="serial",
```

```
79                                 numReadNodes=0,
80                                 mdfpath="/path/to/spde/meta",
81                                 datapath="/path/to/spde/data",
82                                 dfdebug=sqldetails) ;
NOTE: 'MYSPDE' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYSPDE'.
NOTE: Action to ADD caslib MYSPDE completed for session MYSESS1.
83
84    proc casutil ;
NOTE: The UUID '50006a2d-2d64-db42-a993-3f2e06fe30dd' is connected using
session MYSESS1.
85       load  incaslib="myspde" casdata="sparkles"
86            outcaslib="myspde" casout="sparkles" replace;
NOTE: Performing serial LoadTable action using SAS Data Connector to SPDE.
NOTE: Reading Rows from all worker nodes.
NOTE: Up to 360002 rows will be read each by 4 workers for a total of
1440000 rows.
NOTE: Worker #0 will read obs 1 to 360002.
NOTE: Worker #1 will read obs 360003 to 720004.
NOTE: Worker #2 will read obs 720005 to 1080006.
NOTE: Worker #3 will read obs 1080007 to 1440000.
NOTE: Cloud Analytic Services made the external data from sparkles
available as table SPARKLES in caslib myspde.
NOTE: The Cloud Analytic Services server processed the request in 1.440774
seconds.
87    run ;
```

Remember that multi-node data transfer is a feature of SAS Data Connectors (hence the log message about "serial LoadTable action"). Notice also that a new directive was provided in the caslib statement: `dfdebug=sqldetails`. When present, CAS will log additional details about the multi-node data transfer. This debug option is only available for use with select data sources and the results will vary for each.

Caslib srctype	Caslib dataSource Options to Specify Transfer Mode	Data Transfers	Log Shows Number of CAS Workers Transferring Data
SPDE	**dataTransferMode="serial"** **numReadNodes=0** (or >1) **numWriteNodes=0** (or >1)	Multi-node	Yes, when dfdebug=sqldetails
	dataTransferMode="serial" numReadNodes=1 numWriteNodes=1	Serial only	No
	dataTransferMode="parallel"	Parallel only using the SAS EP	No
	dataTransferMode="auto" **numReadNodes=0** (or >1) **numWriteNodes=0** (or >1)	Parallel if possible, else serial	Yes, for multi-node when dfdebug=sqldetails

Table 6. Data Transfer Modes for Filetypes in an SPDE Type of CASLIB

Unlike other CASLIB srcTypes, data transfer defaults to multi-node using the Data Connector for SPD Engine. This makes sense in that SPD Engine files are purpose-built for multi-threaded access. If the SAS In-Database Embedded Process is deployed to Hadoop where the SPD Engine files are stored in HDFS, then fully parallel transfer is also available there.

Minimizing Confusion

Key takeaways from data transfer using the SPD Engine type of CASLIB are as follows:

- The SAS Data Connector for the SPD Engine is included with CAS as a native access mechanism. No additional purchase is necessary, and no associated SAS/ACCESS product is needed.
- Multi-node data transfer is the default. This is unlike other srcTypes which default to serial-only transfer when using SAS Data Connector products
- Use the optional dfdebug=sqldetails parameter to show multi-node transfer with a breakdown of the range of table observations that each CAS worker will load directly.

EXTERNAL DATA SOURCES

CAS can also work with non-native data from external data providers. If the data we want for CAS is not stored in a SAS format, it is usually in a remote system. To get to that data, we need specific software components.

► SAS Data Connector – serial, except when it's parallel

CAS can communicate with external data providers by implementing the appropriate SAS Data Connector product (offered with SAS/ACCESS products). SAS offers Data Connectors for Hadoop, Impala, Oracle, PostgreSQL, ODBC, PC Files, and many more.

The CASLIB syntax varies by data provider to specify details like schema, server, and user credentials.

Serial Transfer with SAS Data Connector

The default mode of operation used by CAS for SAS Data Connectors is serial transfer such that the CAS controller connects to the data source, queries the data, and distributes the data to the CAS workers.

The following code shows sample syntax for serial data transfer using a SAS Data Connector for Postgres:

```
caslib mypg sessref= mysess1
            datasource=(srctype="postgres",
                        username="pguser",
                        password="pgpass",
                        server="pg.site.com",
                        database="mydb",
                        schema="myschema") ;

proc casutil;
   load casdata="polygons" casout="polygons_pg";
quit;
```

Run the code above to get the following SAS log results:

```
78    caslib mypg sessref=mysess1
79                datasource=(srctype="postgres",
80                            username="pguser",
81                            password=XXXXXXXX,
82                            server="pg.site.com",
83                            database="mydb",
84                            schema="myschema";
NOTE: Executing action 'table.addCaslib'.
NOTE: 'MYPG' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYPG'.
NOTE: Action 'table.addCaslib' used (Total process time):
NOTE:       real time               0.021744 seconds
NOTE:       cpu time                0.042650 seconds (196.15%)
NOTE:       total nodes             5 (20 cores)
NOTE:       total memory            156.32G
```

```
NOTE:       memory                  4.15M (0.00%)
NOTE: Action to ADD caslib MYPG completed for session MYSESS1.
85
86    proc casutil;
NOTE: The UUID 'f1fb1e76-65a3-bd4b-8f19-c30a8afbf738' is connected using
session MYSESS1.
87       load casdata="polygons" casout="polygons_pg";
NOTE: Executing action 'table.loadTable'.
NOTE: Performing serial LoadTable action using SAS Data Connector to
PostgreSQL.
NOTE: Cloud Analytic Services made the external data from polygons
available as table POLYGONS_PG in caslib MYPG.
NOTE: Action 'table.loadTable' used (Total process time):
NOTE:       real time               0.117057 seconds
NOTE:       cpu time                0.134396 seconds (114.81%)
NOTE:       total nodes             5 (20 cores)
NOTE:       total memory            156.32G
NOTE:       memory                  17.82M (0.01%)
NOTE:       bytes moved             25.92K
NOTE: The Cloud Analytic Services server processed the request in 0.117057
seconds.
88    quit;
```

Multi-node Transfer with SAS Data Connector

Multi-node transfer occurs when the CAS controller directs each CAS worker to query its allotment of data through its SAS Data Connector (and associated third-party data provider client).

Multi-node transfer is available for most SAS Data Connector products (except JDBC, ODBC, and PC File Formats). Optional caslib dataSource parameters that enable multi-node transfer are as follows:

- Loading data: numReadNodes with a value of 0 or >1
- Saving data: numWriteNodes with a value of 0 or >1

If not specified for external data sources, numReadNodes (and numWriteNodes) parameters will run with a default value of 1, meaning serial-only transfer through the CAS controller.

The following code shows sample syntax for multi-node data transfer using SAS Data Connectors:

```
caslib mypg sessref= mysess1
            datasource=(srctype="postgres",
                        username="myuser",
                        password="mypass",
```

```
                          server="pg.site.com",
                          database="mydb",
                          schema="myschema",
                          numreadnodes=10,numwritenodes=10) ;

proc casutil;
   load casdata="polygons" casout="polygons_pg";
quit;
```

Run the code above to get the following SAS log results:

```
78    caslib mypg sessref=mysess1
79                datasource=(srctype="postgres",
80                            username="casdm",
81                            password=XXXXXXXX,
82                            server="sasdb.race.sas.com",
83                            database="casdm",
84                            schema="public",
85                            numreadnodes=10,numwritenodes=10) ;
NOTE: Executing action 'table.addCaslib'.
NOTE: 'MYPG' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'MYPG'.
NOTE: Action 'table.addCaslib' used (Total process time):
NOTE:       real time               0.021744 seconds
NOTE:       cpu time                0.042650 seconds (196.15%)
NOTE:       total nodes             5 (20 cores)
NOTE:       total memory            156.32G
NOTE:       memory                  4.15M (0.00%)
NOTE: Action to ADD caslib MYPG completed for session MYSESS1.
86
87    proc casutil;
NOTE: The UUID 'f1fb1e76-65a3-bd4b-8f19-c30a8afbf738' is connected using
session MYSESS1.
88       load casdata="polygons" casout="polygons_pg";
NOTE: Executing action 'table.loadTable'.
NOTE: Performing serial LoadTable action using SAS Data Connector to
PostgreSQL.
WARNING: The value of numReadNodes(10) exceeds the number of available
worker nodes(4). The load will proceed with numReadNodes=4.
NOTE: Cloud Analytic Services made the external data from polygons
available as table POLYGONS_PG in caslib MYPG.
```

```
NOTE: Action 'table.loadTable' used (Total process time):
NOTE:       real time               0.117057 seconds
NOTE:       cpu time                0.134396 seconds (114.81%)
NOTE:       total nodes             5 (20 cores)
NOTE:       total memory            156.32G
NOTE:       memory                  17.82M (0.01%)
NOTE:       bytes moved             25.92K
NOTE: The Cloud Analytic Services server processed the request in 0.117057
seconds.
89    quit;
```

Notice in this log that SAS says it's performing a *serial* LoadTable action where multi-node transfer occurs. The word "serial" in this usage is just another way of saying that the SAS Data Connector software is used. The workers do indeed perform a multi-node load where each queries its assigned allotment of data from the source.

There are many requirements to ensure full participation of all designated CAS workers in the transfer. If any of those requirements fail, then CAS will gracefully fallback to fewer CAS workers, all the way down to one if necessary.

By default, the SAS log does not show which CAS workers are participating in the data transfer (or even how many). But we can coax some information out by specifying a value of numReadNodes (and/or numWriteNodes) that is greater than the real number of CAS workers available. Doing so will cause SAS to show a warning message in the log that indicates the number of CAS workers that it will attempt to use.

CASLIB srctypes	Caslib dataSource Options to Enable Multi-node Transfer	Data Transfers	Log Shows Number of CAS Workers Transferring Data
JDBC, ODBC, Path (PC Files)	None	Serial only	No
DB2, HANA, Impala, MySQL, Oracle, Postgres, Redshift, Spark, SQLServer, Vertica	**numReadNodes=1** **numWriteNodes=1**	Serial only	No
	numReadNodes=0 (or >1) numWriteNodes=0 (or >1)	Multi-node	When numReadNodes is greater than number of CAS workers
Hadoop, Teradata	**dataTransferMode="serial"** **numReadNodes=1** **numWriteNodes=1**	Serial only	No
	dataTransferMode="serial" numReadNodes=0 (or >1) numWriteNodes=0 (or >1)	Multi-node	When numReadNodes is greater than number of CAS workers

Table 7. Data Transfer Modes for Sources Using a SAS Data Connector

Notice that for most CASLIB srcTypes, the dataTransferMode option cannot be specified. In these cases, it is effectively "serial", meaning that the appropriate SAS Data Connector technology is used. Also, the default value for numReadNodes and numWriteNodes for SAS Data Connectors to third-party data sources is "1", meaning that the CAS controller will serially load the data from the source and distribute it to the workers. Multi-node data transfer can be enabled by setting numReadNodes and/or numWriteNodes to "0" (meaning all CAS workers) or to a value greater than "1" (to specify a limited number of CAS workers). JDBC, ODBC, and Path (for PC Files) srcTypes cannot use multi-node data transfer.

Minimizing Confusion

Key takeaways from data transfer using the SAS Data Connectors are as follows:

- CASLIBs defined to external data sources default to using the associated SAS Data Connector software (that is, if available, dataTransferMode="serial").

- When dataTransferMode is specified in the caslib dataSource options, the value "serial" actually means "Use SAS Data Connector software", which includes the ability to perform multi-node transfer (which, under ideal conditions, is effectively parallel movement).
- The dataTransferMode can only be specified for the external data sources that SAS offers SAS Data Connect Accelerator software: Hadoop, Spark, and Teradata as of SAS Viya 3.4. Otherwise, the value is effectively "serial" (meaning "Use SAS Data Connector software") and therefore multi-node transfer is possible.
- Multi-node data transfer is enabled using the numReadNodes and numWriteNodes caslib dataSource options. Multi-node transfer is not available for JDBC, ODBC, or PC Files.
- The value of numReadNodes and numWriteNodes is a *suggestion*, not a requirement. CAS will attempt to use the number of nodes specified. However, if it can only proceed by using fewer nodes, then it will.
- Only the CAS workers that perform the data transfer will have the data available in memory. If that data should be distributed across the remaining workers as well, then additional steps are needed to make it happen – it's not automatic.
- If only one data connection from CAS is possible (or numReadNodes=1), then standard serial transfer will occur where the CAS controller loads the data from source and distributes it across all workers.
- For more information about which CAS workers participated in the multi-node transfer, refer to the Recommended Reading section below for the web article titled "SAS® Data Connector/SAS® Data Connect Accelerator Logging".

► SAS Data Connect Accelerator – perfectly parallel, with fallback to serial

SAS currently offers Data Connect Accelerator (DCA) products to work with Hive and Spark in Hadoop as well as Teradata. The DCA works in conjunction with our SAS® In-Database technology in the form of the SAS® Embedded Process (EP). The EP is a specialized software component that is deployed inside the remote data provider. Its primary purpose is to "take the analytics to the data" to reduce data movement and duplication. SAS offers data quality, scoring, and code acceleration functionality with our SAS In-Database products.

To access the full power of products like SAS® Visual Analytics, SAS® Visual Statistics, and SAS® Visual Data Mining and Machine Learning, data must be brought over into CAS. And the EP can aid with data transfer to CAS as well. The SAS Embedded Process for SAS Viya 3.4 can transfer data in parallel to CAS from Hive tables, Spark tables, SAS SPD Engine tables in HDFS, and Teradata tables.

Each instance of the EP in a data provider can be directed to send its allotment of rows to all CAS workers ensuring even distribution of the data. When directed, this is a parallel transfer of data with all CAS workers participating.

The following code shows sample syntax for parallel data transfer using the SAS Data Connect Accelerator:

```
caslib cashive datasource=(srctype="hadoop",
                           server="hadoop.site.com",
                           username="hadoopuser",
                    dataTransferMode="parallel",
                    hadoopconfigdir="/opt/sas/viya/config/data/hadoop/conf",
                    hadoopjarpath="/opt/sas/viya/config/data/hadoop/lib",
```

```
                              schema="hiveschema"
                              dfdebug=epinfo);

proc casutil ;
   load incaslib="cashive" casdata="endeavors"
        outcaslib="cashive" casout="endeavors" replace ;
run ;
```

Run the code above to get the following SAS log results:

```
85    /* Define the caslib for parallel execution */
86    caslib cashive datasource=(srctype="hadoop",
87                               server="hadoop.site.com",
88                               username="hadoopuser",
89                               dataTransferMode="parallel",
90
hadoopconfigdir="/opt/sas/viya/config/data/hadoop/conf",
91
hadoopjarpath="/opt/sas/viya/config/data/hadoop/lib",
92                               schema="cashive"
93                               dfdebug=epinfo);
NOTE: 'CASHIVE' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'CASHIVE'.
NOTE: Action to ADD caslib CASHIVE completed for session MYSESS1.
94
95    proc casutil ;
NOTE: The UUID '5bfbf791-adbd-1d47-81d0-7da43c1964d6' is connected using
session MYSESS1.
96       load  incaslib="cashive" casdata="endeavors"
97            outcaslib="cashive" casout="endeavors" replace ;
NOTE: Performing parallel LoadTable action using SAS Data Connect
Accelerator for Hadoop.
NOTE: SAS Embedded Process tracking URL:
http://sashdp01.race.sas.com:8088/proxy/application_1547927219062_0045/
NOTE: Job Status ......: SUCCEEDED
NOTE: Job ID ..........: 1547927219062_45
NOTE: Job Name ........: SASEP SuperReader
/apps/hive/warehouse/cashive.db/endeavors
NOTE: File splits..... : 2
NOTE: Input records ...: 322
NOTE: Input bytes .....: 37048
```

```
NOTE: Output records ..: 0
NOTE: Output bytes ....: 0
NOTE: Transcode errors : 0
NOTE: Truncations .....: 0
NOTE: Map Progress ....: 100.00%
NOTE: Cloud Analytic Services made the external data from endeavors
available as table ENDEAVORS in caslib cashive.
NOTE: The Cloud Analytic Services server processed the request in 27.066701
seconds.
98    run ;
```

The dataSource option "dfdebug=epinfo" shows additional information and statistics in the log – of interest is the tracking URL to monitor the SAS Embedded Process job's execution in YARN. This debug option has no impact on execution efficiency. However, if you need additional help with troubleshooting an issue, then "dfdebug=all" can be used to enable the generation of more details by the SAS Embedded Process. Take care, however, as that option does more work, increasing the runtime of the jobs and generating larger log files.

Caslib srctype	Caslib dataSource Options to Enable Parallel Transfer	Data Transfers	Log Shows Transfer Status
Hadoop	`dataTransferMode="auto"`	Parallel using DCA, if possible. Else serial using DC	Yes and more using `dfdebug=epinfo`
	`dataTransferMode="parallel"`	Parallel using DCA only	Yes and more using `dfdebug=epinfo`
Teradata	`dataTransferMode="auto"`	Parallel using DCA, if possible. Else serial using DC	Yes and more using `dfdebug=epinfo`
	`dataTransferMode="parallel"`	Parallel using DCA only	Yes and more using `dfdebug=epinfo`
SPDE	`dataTransferMode="auto"`	Parallel using DCA, if possible. Else serial using DC	Yes and more using `dfdebug=sqldetails`
	`dataTransferMode="parallel"`	Parallel using DCA only	No

Table 8. Data Transfer Modes for Sources Using a SAS Data Connect Accelerator with the SAS In-Database Embedded Process

Minimizing Confusion

Key takeaways from data transfer using the SAS Data Connect Accelerator are as follows:

- When dataTransferMode is specified in the caslib dataSource options, the value "parallel" means "Use SAS Data Connect Accelerator software". All CAS workers participate in "parallel" transfer, receiving their allotments of data from every instance of the SAS Embedded Process job running in the data provider.
- When dataTransferMode is specified in the caslib dataSource options, the value "auto" means to try "parallel" (that is, using the SAS Data Connect Accelerator software), else try "serial" (that is, using the SAS Data Connector software).

- "Auto" is not the default value for dataTransferMode in the caslib dataSource option. "Serial" (that is, using the SAS Data Connect software) is the default.
- For "auto", if "parallel" fails and "serial" is attempted, then CAS will check for the numReadNodes and numWriteNodes dataSource options in the caslib and attempt multi-node transfer as suggested.
- If the suggested number of CAS workers cannot participate in the multi-node transfer, CAS will automatically proceed with those that can.
- If only one CAS host can perform the transfer (or numReadNodes=1), then standard serial transfer will occur where the CAS controller loads the data from source and distributes to the CAS workers.

- The SAS log mentions "serial" and "parallel" LoadTable actions to inform whether the SAS Data Connector product or SAS Data Connect Accelerator product, respectively, was used for the data transfer. "Serial" is shown when multi-node transfer occurs.
- Refer to the Recommended Reading section below for the web article titled "SAS® Data Connector/SAS® Data Connect Accelerator Logging" for more information.

EXTERNAL DATA SOURCES: GRACEFUL FALLBACK FROM PERFECTLY PARALLEL TO SERIOUSLY SERIAL

In many cases, CAS can perform parallel data transfer using the SAS Data Connect Accelerator software alongside the SAS Embedded Process. If there is a problem, then CAS can be directed to attempt multi-node data transfer using SAS Data Connector software instead. If multi-node cannot be achieved, then CAS will try serial transfer next.

The following code shows sample syntax for auto determining data transfer:

```
caslib cashive datasource=(srctype="hadoop",
                           server="hadoop.site.com",
                           username="hadoopuser",
                    dataTransferMode="auto",
                    hadoopconfigdir="/opt/sas/viya/config/data/hadoop/conf",
                    hadoopjarpath="/opt/sas/viya/config/data/hadoop/lib",
                           schema="hiveschema",
                           numreadnodes=10,numwritenodes=10);
```

Specifying dataTransferMode="auto" as a caslib dataSource option will allow CAS to do the following:

1. First try parallel transfer:
   Use the SAS Data Connect Accelerator software to communicate with the SAS Embedded Process in the remote data provider for a massively parallel data transfer.

   If successful, the SAS log will show the following:

   ```
   NOTE: Performing parallel LoadTable action using SAS Data Connect
   Accelerator for Hadoop.
   ```

   Parallel transfer has best performance for largest tables and many concurrent jobs.

2. Else try multi-node transfer:
   Use the SAS Data Connector software to communicate with the remote data provider through the local data client software on 2 or more CAS workers if the numReadNodes and/or numWriteNodes parameters are set to 0 or a value greater than 1.

   If successful, the SAS log will show the following:

   ```
   NOTE: Performing serial LoadTable action using SAS Data Connector for
   Hadoop.
   ```

   Multi-node transfer has better performance for large tables and limited concurrent jobs. It supports a wide range of data sources and file types.

   Remember to specify a value for numReadNodes and/or numWriteNodes that is greater than the actual number of CAS workers to see a note in the SAS log that indicates CAS is aware of multi-node capability.

3. Else try serial-only transfer:
   Use the SAS Data Connector software to communicate with the remote data provider through the local data client software on the CAS controller. The controller reads all data and distributes it to the workers.

   If successful, the SAS log will show the following:

   ```
   NOTE: Performing serial LoadTable action using SAS Data Connector for
   Hadoop.
   ```

   Serial only transfer is available for the widest range of data sources and file types.

Take care with the fully automatic approach from external data sources. For very large tables where serial-only loading will take an excessive amount of time to perform, consider specifying dataTransferMode="parallel" (when possible) to ensure that a very long job is not run unexpectedly.

Also remember that native caslib srcTypes DNFS, HDFS, S3, and SPDE are available for parallel (or at least multi-node) loading of SASHDAT and CSV files. Those srcTypes do not require licensing additional software like SAS/ACCESS, SAS Data Connectors to external data sources, SAS Data Connect Accelerators, or SAS In-Database Embedded Process.

TERMINOLOGY TAKEAWAYS

It's said that the native peoples of the North American Arctic region have over 50 words for "snow". Unfortunately, when it comes to describing data transfer modality and SAS programming syntax, we don't have that much variety of synonyms to use to describe how data moves. This means that some terms are overloaded with multiple meanings.

- Serial
  1. *General use*: Data moves sequentially along a single path.
  2. *For MPP CAS*: The CAS controller loads the data from source, and then distributes it to the CAS workers. Saving back to source occurs in reverse order.
  3. *SAS programming syntax*:
     - caslib > dataSource > dataTran**sferMode="serial"**: Use the associated SAS Data Connector technology to access the data source. The resulting

transfer may be serial (i.e. CAS Controller reads from source, then distributes) or multi-node, as directed.

- PROC > importOptions > dataTransferMod**e="serial"**: The CAS controller loads the data from source, and then distributes it to the CAS workers. Saving back to source occurs in reverse order.

- Parallel
  1. *General use*: Data moves along multiple, concurrent paths at the same time.
  2. *For MPP CAS*: The CAS controller directs each of the CAS workers to access their allotment of data directly from the source.
  3. Massively parallel: For some data sources, CAS can direct its workers to each use multiple threads when accessing data. This potentially multiplies the transfer rate yet again and aids in distribution of data.
  4. *SAS programming syntax*:
     - caslib > dataSource > **dataTransferMode="parallel"**: Use the associated SAS Data Connect Accelerator technology to communicate with the SAS In-Database Embedded Process to access the data source.
     - PROC > importOptions > **dataTransferMode="parallel"**: The CAS controller directs each of the CAS workers to access their allotment of data directly from the source.
- SAS Data Connector
  1. *Technology:* CAS accesses data through the use of data connectors. Data connectors are an extensible technology that allow SAS to modularly add the ability to access an ever-growing range of native and external data sources.
  2. *Product:* SAS offers additional, optional software products to a range of third-party data sources. Typically, SAS/ACCESS products for SAS Viya include an associated SAS Data Connector product to enable CAS to directly access data in a third-party data source.
- SAS Data Connect Accelerator
  1. *Product:* A SAS Data Connect Accelerator enables CAS to communicate directly with the SAS In-Database Embedded Process deployed to an external data source. Pre-requisites include licensing SAS In-Database, SAS/ACCESS with associated SAS Data Connectors, and installing the external data source's client software on the CAS controller(s).

In addition to words that are overloaded with multiple definitions, some terms used by SAS are unique in meaning:

- Multi-node data transfer
  SAS/ACCESS Data Connector technology enables CAS to perform multi-node data transfer. When directed to perform a multi-node transfer, the CAS controller gets statistics describing the table from the data provider. If that table has a suitable numeric variable, then the controller will direct each CAS worker to query a specific set of rows from the data provider.

  The term "multi-node" is used instead of "parallel" to differentiate based on the possibility that the data provider might have limited capacity on the back end for

concurrent data streams.

If multi-node is attempted, but only one CAS host can actually access the data, then serial data transfer by the CAS controller is performed.

- Distributed Network File System (DNFS)
    1. *Technology*: Refers to the generalized concept of distributed network file system – known as shared file systems and/or clustered file systems. The concept is meant broadly as SAS is typically agnostic about which specific storage solution vendors are used to host data files. For MPP CAS, the idea is that Portable Operating System Interface (POSIX) compliant disk storage is mounted identically at the same path on all CAS hosts.

       Simple NFS will work but is generally known not to perform at scale. To achieve high throughput for fastest loading of data into CAS, consider implementing a high-performance storage solution.
    2. *SAS data format*: CAS can save SASHDAT table data in a single file (internally described simply as a DNFS container) on standard disk. This structure is fundamentally different than storing SASHDAT table data in Hadoop Distributed File Systems (HDFS) where it's represented as many discrete files.
    3. *SAS programming syntax*:
       caslib > dataSource > srcType="dnfs": Directs CAS to access SASHDAT (or comma-separated values (CSV) files) from the shared file system mounted identically on all CAS hosts. The CAS controller directs each CAS worker to load (or save) its allotment of data from the SASHDAT file.

DATA TRANSFER NOT DISCUSSED HERE

This paper provides a brief glimpse at data movement concepts with SAS Cloud Analytic Server in SAS Viya 3.4 for many data sources, but not all of them. There are additional sources where data is stored that can also provide data for CAS (Figure 8).

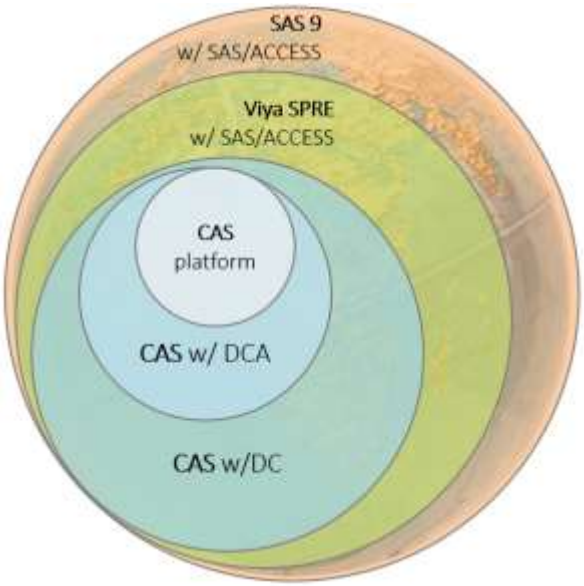

Figure 8. CAS Has Access to a World of Data

SAS PROGRAMMING RUNTIME ENVIRONMENT

SAS Viya offers the SAS Programming Runtime Environment for the execution of custom SAS program code. The code examples in this paper were developed using SAS Studio and the SAS Programming Run-time Environment components, like the SAS Viya Compute Server. Any data that is accessible to the SAS Viya Compute Server can be sent to CAS (that is, to the CAS controller for subsequent distribution to the CAS workers).

The SAS Viya Compute Server provides the classic SAS run time for running SAS program code. It can be extended to work with additional external data sources by licensing the necessary SAS/ACCESS products. Further, there are some SAS/ACCESS products for the SAS Viya Compute Server that do not have equivalent SAS Data Connectors, such as HAWQ, Netezza, Greenplum, SAP ASE (formerly Sybase), and SAP R/3. With the power of original SAS on hand, native formats like SAS data sets, catalogs, SPD Engine tables, and more are available to work with as well.

SAS 9.4

If SAS 9.4 is also available in the environment, then the list of SAS/ACCESS engines is even longer. They include those already mentioned in the previous section as well as ADABAS, Aster, CA IDMS TM, SAP IQ, DATACOM/DB, IMS-DL/I, INFORMIX, OLEDB, SYSTEM 2000, and PI System.

MOVING DATA FROM SAS TO CAS

Whichever SAS execution is available, use the recently introduced "cas" LIBNAME engine to serially transfer data directly over for CAS to work with. If working with an older version of SAS, then SAS/CONNECT technology can act as a bridge to transfer data serially as well. There are no parallel transfer capabilities between the SAS runtimes and CAS.

Another approach is to rely on a shared file system technology which is available to both SAS and CAS. This allows SAS to read and write its data files to the same location which CAS could use the PATH or DNFS types of CASLIBs to load data in parallel, as appropriate.

CAS_DISK_CACHE

CAS was built to offer more resilience and robustness than LASR in countering the problems of hardware failure. One way it does this is to rely on a disk-based backing store for in-memory data – the CAS_DISK_CACHE. The idea is that, if needed, each CAS worker can recover data from the CAS_DISK_CACHE. This is similar in concept to a parallel data load since the CAS workers each re-load their specific allotment of data from local disk.

The workings and considerations of the CAS_DISK_CACHE are beyond the scope of this paper. Suffice to say, if large volumes of data must be fetched from CAS_DISK_CACHE frequently, then consider ensuring that the storage solution providing that disk is optimized appropriately. At a high level, ensure that every CAS worker host has local disk(s) set aside to provide the CAS_DISK_CACHE space. Avoid shared and clustered file system technologies. Local hardware RAID for speed and redundancy is expected to be sufficient but is not prescribed as the sole option.

For a detailed look at these considerations, refer to *Engineering CAS Performance Hardware: Network, and Storage Considerations for CAS Servers* by Tony Brown, presented at SAS Global Forum 2019 as Paper 3351-2019.

SAS VISUAL INTERFACES

SAS offers many point-and-click visual interfaces for reporting, analytics, modeling, management, as well as data cleansing and transfer. Each of those products leverages CAS

abilities as suits their purpose. In many cases, the visual interface gives little or no indication as to *how* CAS accessed the data (serial, multi-node, or parallel) – just that the action succeeded or not.

Fortunately, the CAS nodes write log files to disk in their respective installation directories. Refer to those logs, possibly enabling additional log tracing options as directed by SAS Technical Support, to get more detailed information.

CLOUD DATA EXCHANGE

Cloud Data Exchange is a recent feature offered with SAS Viya. It provides the ability to securely copy data between on-premises sources and cloud-based sources. Instead of CAS communicating directly with the remote data source, the SAS® Data Agent is used in the middle to facilitate the transfer. Cloud Data Exchange can implement forms of serial and parallel data movement.

One enhancement made with CDE is that it ensures data is evenly distributed (or as partitioned) to all session CAS workers automatically. There's no extra step as mentioned for multi-node transfer when only a subset of CAS workers participate in the transfer.

CONCLUSION

As data volumes continue to increase, the ability to load data quickly at scale is an important goal. Providing the infrastructure in support of scalable data loading using multiple, concurrent, parallel channels requires planning, expenditure, and ongoing support. To ensure maximum value of those investments, it is useful to validate that data is moving from source to destination as quickly and efficiently as intended.

SAS software is flexible to meet the varied requirements of accessing data from a myriad of native and external data sources. The SAS Cloud Analytic Server is purpose-built to achieve massive parallel performance, not just in analytics, but in the movement of data as well.

The syntax to accomplish this data movement and to understand it has evolved quickly, but not consistently across all areas. Familiarization with the nuances of how data moves for the various data sources is necessary to ensuring that proper data transfer is achieved.

RECOMMENDED READING

SAS Documentation:

- *Support for Databases in SAS® Viya® 3.4*
- See *SAS® Cloud Analytic Services 3.4: Fundamentals* for the following:
    - Architecture
    - Data life cycle
    - How SAS Cloud Analytic Services Uses Memory
    - What is a Caslib?
- See *SAS® Cloud Analytic Services 3.4: User's Guide* for the following:
    - CAS statement
    - CAS LIBNAME statement
    - CASLIB statement

    - CAS procedure
    - CASUTIL procedure
    - Platform Data Sources
    - Data Connectors and Data Connect Accelerators
- See *SAS® Viya® 3.4 for Linux: Deployment Guide* for the following:
    - SAS In-Database Technologies Embedded Process
    - Verifying SAS Data Connect Accelerator

Web articles:

- *Understanding CAS: 6 Key CAS Data Concepts*
  by Stephen Foerster, 17-AUG-2018, SAS Communities Library
  communities.sas.com/t5/SAS-Communities-Library/Understanding-CAS-6-Key-CAS-Data-Concepts/ta-p/487776
- *SAS® Data Connector/SAS® Data Connect Accelerator Logging*
  by Nicolas Robert, 19-SEP-2018, SAS Communities Library
  communities.sas.com/t5/SAS-Communities-Library/SAS-Data-Connector-SAS-Data-Connect-Accelerator-Logging/ta-p/496832
- *Accessing SPDE files with CAS*
  by Nicolas Robert, 26-OCT-2018, SAS Communities Library
  communities.sas.com/t5/SAS-Communities-Library/Accessing-SPDE-files-with-CAS/tac-p/508614#M3277
- *SAS® Cloud Data Exchange Preview*
  by Uttam Kumar, 14-DEC-2018, SAS Communities Library
  communities.sas.com/t5/SAS-Communities-Library/SAS-Cloud-Data-Exchange-Preview/ta-p/521657

PROFESSIONAL APPRECIATION

The details shared in this paper would not have been possible without the generous input and deep expertise from many folks who volunteered to help. Special thanks to my SAS colleagues: Brian Bowman, Lisa Brown, Jeff Cleveland, Stephen Foerster, Mark Gass, Bill Hefner, Steve Krueger, Uttam Kumar, Nicolas Robert, Jason Secosky, Scott Vance, and Dave Zanter.

CONTACT INFORMATION

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

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