

SAS2560-2016

Ten Tips to Unlock the Power of Hadoop with SAS®

Wilbram Hazejager and Nancy Rausch, SAS Institute Inc.

ABSTRACT

This paper discusses a set of practical recommendations for optimizing the performance and scalability of your Hadoop system using SAS®. Topics include recommendations gleaned from actual deployments from a variety of implementations and distributions. Techniques cover tips for improving performance and working with complex Hadoop technologies such as Kerberos, techniques for improving efficiency when working with data, methods to better leverage the SAS in Hadoop components, and other recommendations. With this information, you can unlock the power of SAS in your Hadoop system.

INTRODUCTION

When traditional data storage or computational technologies struggle to provide either the storage or computation power required to work with large amounts of data, an organization is said to have a big data issue. Big data is frequently defined as the point at which the volume, velocity, and/or variety of data exceeds an organization's storage or computation capacity for accurate and timely decision-making.

The most significant new technology trend that has emerged for working with big data is Apache™ Hadoop®. Hadoop is an open source set of technologies that provide a simple, distributed storage system paired with a fault tolerant parallel processing approach that is well suited to commodity hardware. Many organizations have incorporated Hadoop into their enterprise leveraging the ability for Hadoop to process and analyze large volumes of data at low cost.

SAS has extensive integration options with Hadoop to bring the power of SAS to help address big data challenges. SAS, via the SAS/ACCESS® technologies and SAS® In-Database Code Accelerator products, has been optimized to push down computation and augment native Hadoop capabilities to bring the power of SAS to the data stored in Hadoop. By reducing data movement, processing times decrease and users are able to more efficiently use compute resources and database systems.

The following recommendations describe some of the best practices to help you make the most of your SAS and Hadoop integration.

TIP #1: PARALLEL DATA LOADS

When loading data from a database into your Hadoop cluster, there are many things to consider. One of them is the network layout, especially if you need to transfer lots of data. Here is an example of a network layout that is quite common, showing a dedicated high-bandwidth network connection between relational database management System (RDBMS) and Hadoop cluster, with the SAS server on a slower network:

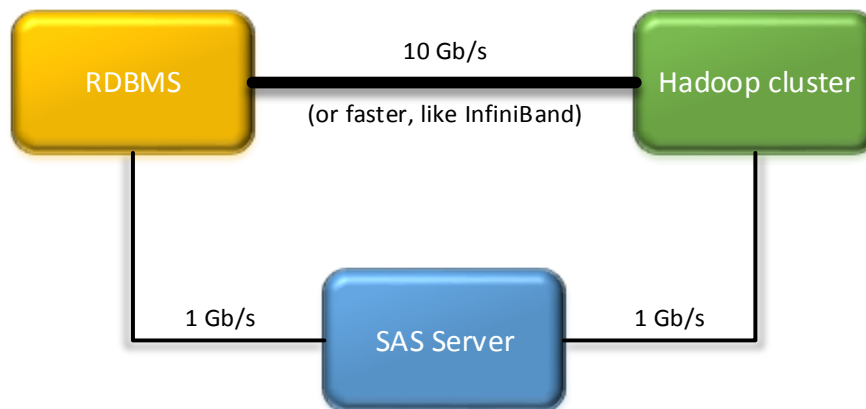


Figure 1: Network Layout

To transfer data between database and Hadoop, depending on whether you prefer to write DATA step code or SQL code, you could use one of the following (simplified) approaches:

```
data hdfsplib.table_out;
  set oralib.table_in;
run;
```

or

```
proc sql;
  create table hdfsplib.table_out
  as select * from oralib.table_in;
run;
quit;
```

When using this approach, you can take advantage of the full breadth of SAS functions to transform the data while it is being transferred from your database to Hadoop. However, all of the data has to pass through the SAS server and therefore has to travel over the slower 1 Gb/s network.

To take advantage of the faster network you can use PROC SQOOP. This new procedure has been introduced in the third maintenance release of SAS 9.4 and is used to access Apache Sqoop™ from a SAS session to directly transfer data in both directions between a database and the Hadoop Distributed File System (HDFS). Basically this SAS procedure initiates the transfer of data. Sqoop runs on your Hadoop cluster and directly connects to the database, so the data travels on the high-bandwidth network.

Sqoop commands are passed to the Hadoop cluster using the Apache Oozie Workflow Scheduler for Hadoop. PROC SQOOP defines an Oozie workflow that includes a Sqoop task, which is then submitted to an Oozie server using a RESTFUL API.

PROC SQOOP works similarly to the Apache Sqoop command-line interface, using the same syntax. The procedure provides feedback as to whether the job completed successfully and where to get more details in your Hadoop cluster.

Your Hadoop administrator needs to have Sqoop configured for use with the database, meaning a database provider's JDBC driver needs to be made available to the Oozie/Sqoop installation.

PROC SQOOP uses your Hadoop cluster configuration files, just like the SAS/ACCESS® Interface to Hadoop functionality and the procedure is available on Windows and UNIX. More information about supported Hadoop distributions can be found on this page:

<http://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoop-distributions.html>.

To use PROC SQOOP you must have the following information: the JDBC connection string for your database, credentials for both Hadoop and database, and the following Hadoop configuration information: URLs for Oozie, NameNode service, and JobTracker.

The following code examples illustrates the data transfer from a Teradata database to HDFS while selecting a subset of the source data:

```
options set=SAS_HADOOP_CONFIG_PATH="/mydir/Hadoop_sitexmls";
options set=SAS_HADOOP_RESTFUL=1;
proc sqoop dbuser='mydbusr1' dbpwd='mydbpwd1'
  hadoopuser='sashdpusr1' hadooppwd='sashdppwd1'
  oozieurl='http://xxxx.yyyy.com:11000/oozie'
  namenode='hdfs://xxxx.yyyy.com:8020'
  jobtracker='xxxx.yyyy.com:8032'
  wfhdspath='hdfs://xxxx.yyyy.com:8020/user/mydbusr1/myworkflow.xml'
deletewf
command='import
--connection-manager com.cloudera.connector.teradata.TeradataManager
--connect jdbc:terahost.yyyy.com/database=myteradb
--query "SELECT * FROM sales where ($CONDITIONS and I < 25)" -m 1
--split-by i
```

```

--delete-target-dir
--target-dir /user/mydbusr1/sales2';
run;

```

Note that the above shown connection-manager option is for Cloudera. Please refer to the documentation of your Hadoop distribution for more details.

For more information about Apache Sqoop, see the online documentation at <http://sqoop.apache.org>. For more information about PROC SGOOP options, see the *Base SAS® 9.4 Procedures Guide, Fifth Edition*.

You can also use Sqoop from SAS® Data Loader for Hadoop. The Copy Data to Hadoop directive copies data from a database to your Hadoop cluster using Sqoop. A connection to the database is defined using a configuration screen:

The screenshot shows a 'Database Configuration' dialog box with the following fields and values:

- Name: * Teradata SGF
- Description: (empty text area)
- Server Configuration:
 - Type: Teradata (dropdown menu)
 - Driver class: * com.teradata.jdbc.TeraDriver
 - Host: * terahost.yyyy.com
 - Database name: * myteradb
 - Connection manager: com.cloudera.connector.teradata
- Connect string: * jdbc:teradata://terahost.yyyy.com/Database=myteradb (with copy, paste, and help icons)
- Authentication:
 - User ID: mydbusr1 (with delete icon)
 - Password: (masked with dots)

Buttons at the bottom: Test Connection, OK, Cancel.

Figure 2: SAS® Data Loader for Hadoop - Database Configuration

Once this connection has been defined, the Copy Data to Hadoop directive allows you to navigate through the tables in your database, select the source table, preview its content, and specify where the target table should be stored in Hive and in which format. There are also options to apply filters and subset the list of columns.

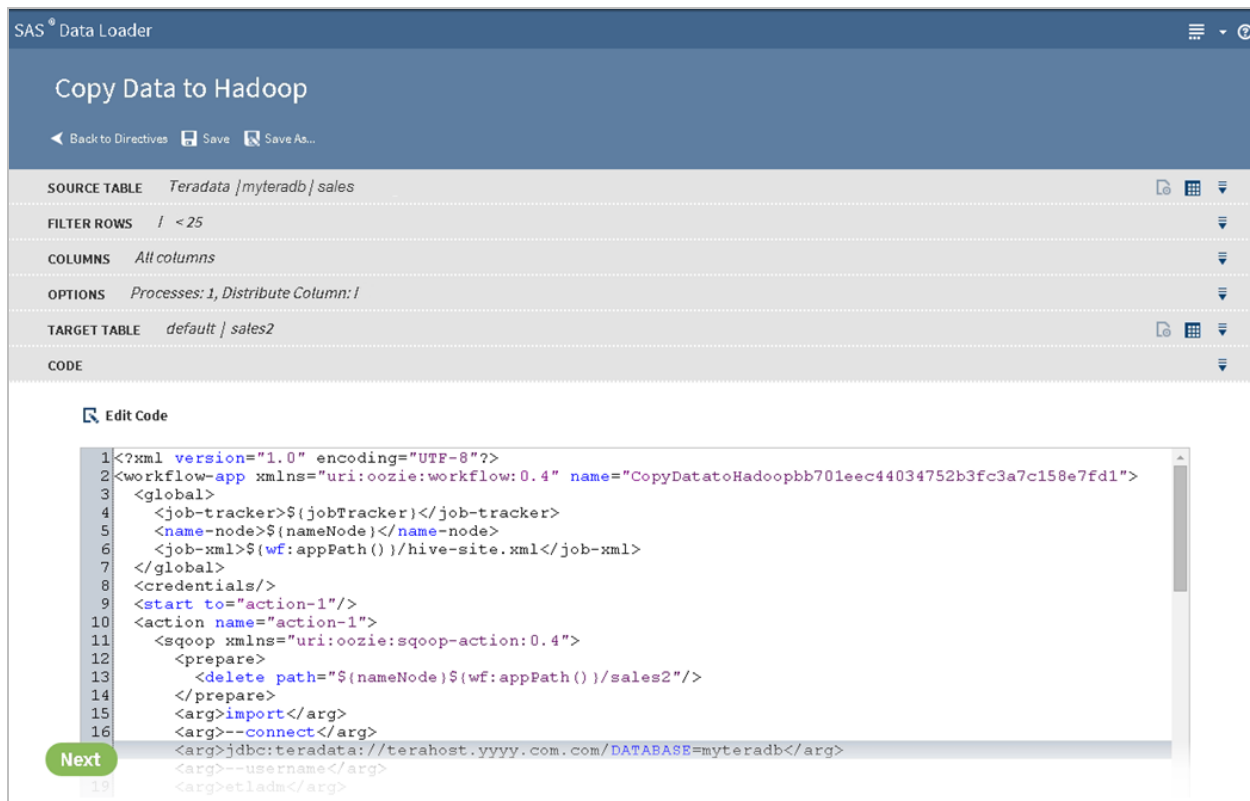


Figure 3: SAS Data Loader for Hadoop - Copy Data to Hadoop Directive

When using SAS Data Loader for Hadoop, the SAS Data Loader mid-tier passes Sqoop commands to the Hadoop cluster using the Apache Oozie Workflow Scheduler for Hadoop. The generated Oozie workflow includes a Sqoop task, which is then submitted to an Oozie server using a RESTFUL API.

TIP #2: DEBUGGING TIPS

If your SAS program that interacts with Hadoop does not work as expected, you can add the SASTRACE option to your SAS code to specify that the SAS log should contain more details about what goes on under the covers.

From the more detailed SAS log you can see what operations are passed down to Hadoop, for example, whether functions, filters, or joins are executed inside Hadoop or whether the data is pulled down to the SAS server. The SASTRACE option is described in detail in the following documentation: *SAS/ACCESS 9.4 for Relational Databases: Reference, Seventh Edition*. That documentation also has a separate chapter for Hadoop that has more information about what operations can be pushed down to Hadoop when using SAS/ACCESS Interface to Hadoop.

Here is a quick code example:

```

options sastrace=',,,d' sastrace=saslog nostsuffix;
libname mydata HADOOP host='xxxx.yyyy.com' port=10000; /* HiveServer2 */
proc sql;
  create table mydata.contacts_nv
  as select * from contacts
  where upcase(state) = 'NV'
  ;
run;quit;
options sastrace=off;

```

and a snippet from the SAS log:

```

16 proc sql;
17     create table mydata.contacts_nv
18     as select id, contact, state
19     from mydata.contacts
20     where upcase(state)='NV';
... text deleted ...
HADOOP_16: Prepared: on connection 2
CREATE TABLE sasdata_14_13_27_149_00001 ROW FORMAT DELIMITED FIELDS
TERMINATED BY '1' LINES TERMINATED BY '10' STORED AS TEXTFILE
LOCATION '/tmp/sasdata_14_13_27_149_00001' AS SELECT `CONTACTS`.`id`,
`CONTACTS`.`contact`, `CONTACTS`.`state`
FROM `CONTACTS` WHERE (UPPER( `state`) = 'NV' )
... text deleted ...

```

This is just a snippet from the SAS log to show that the query was executed by Hive, including the filter.

On a side note: Actually when the SAS/ACCESS Interface to Hadoop loads data to Hive in text file format, it performs the following 3 steps:

- Issues a CREATE TABLE Hive command to specify all table metadata (including column definitions)
- Uses HDFS to upload table data to HDFS
- Issues a LOAD DATA Hive command that moves the data to the appropriate Hive warehouse location

More details can be found in the SAS/ACCESS Interface to Hadoop chapter of the following documentation: *SAS/ACCESS 9.4 for Relational Databases, Reference, Seventh Edition*.

Now back to looking at logs. Sometimes, especially when Hadoop reports errors, the SAS log might not have been able to capture all the details from Hadoop. In that situation you can look at the Hadoop Job History Viewer to browse (a lot more) details about the underlying MapReduce job that was executed. If you have access to the cluster manager, like Cloudera Manager or Ambari, then you'll find links to the Job History Viewer under the MapReduce2/YARN service. If you have access to HUE, then you will find an icon for the Job Browser there. Both webUIs show a list of jobs and you can select your job and look at the various logs. If you do not have access to these, then ask your Hadoop Administrator for the web address of the Job History Viewer in your installation.

In our example proc SQL passed down the query to HiveServer2, so you will see a MapReduce job created by Hive. In the Job History Viewer below you will recognize (part of) the Hive create table statement that was displayed in the SAS log when the SASTRACE option was used:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
2016.03.08 14:13:27 EST	2016.03.08 14:13:32 EST	2016.03.08 14:13:41 EST	job_1455302249427_0135	CREATE TABLE sasdata_14_13_27_149_00001 ... (Stage)	etlguest	root.etlguest	SUCCEEDED	1	1	0	0
2016.03.08 14:09:08 EST	2016.03.08 14:09:13 EST	2016.03.08 14:09:21 EST	job_1455302249427_0134	CREATE TABLE sasdata_14_09_07_232_00001 ... (Stage)	etlguest	root.etlguest	SUCCEEDED	1	1	0	0
2016.03.08 14:02:34 EST	2016.03.08 14:02:39 EST	2016.03.08 14:02:47 EST	job_1455302249427_0133	CREATE TABLE sasdata_14_02_33_269_00001 ... (Stage)	etlguest	root.etlguest	SUCCEEDED	1	1	0	0
2016.03.08 13:59:30 EST	2016.03.08 13:59:36 EST	2016.03.08 13:59:44 EST	job_1455302249427_0132	CREATE TABLE sasdata_13_59_30_029_00001 ... (Stage)	etlguest	root.etlguest	SUCCEEDED	1	1	0	0
2016.03.08 13:55:43 EST	2016.03.08 13:55:48 EST	2016.03.08 13:55:56 EST	job_1455302249427_0131	SAS Map/Reduce Job	etlguest	root.etlguest	SUCCEEDED	2	2	0	0
2016.03.08 13:53:29 EST	2016.03.08 13:53:34 EST	2016.03.08 13:53:43 EST	job_1455302249427_0130	SAS Map/Reduce Job	etlguest	root.etlguest	SUCCEEDED	2	2	0	0

Figure 4: Hadoop Job History UI

When using the SAS® In-Database Code Accelerator for Hadoop, proc DS2 will generate a job named “SAS Map/Reduce Job”. Starting with the third maintenance release of SAS 9.4, when the MSGLEVEL=I option is set and a job fails, a link to the HTTP location of the MapReduce logs is also produced. Here is an example.

```
ERROR: Job job_1424277669708_2919 has failed. Please, see job log for
details. Job tracking URL :
http://xxxx.yyyy.com:8088/proxy/application_1424277669708_2919/
```

The HTTP link is to a site that contains the job summary. On the job summary page, you can see the number of failed and successful tasks. If you click on the failed tasks, you see a list of task attempts. A log is assigned to each attempt. Once in the log page, you are able to see the error messages.

TIP #3: TAKE ADVANTAGE OF YOUR SQL ENGINE

Your Hadoop distribution might come with multiple SQL engines that are supported by SAS. The third maintenance release SAS 9.4 supports Hive using SAS/ACCESS Interface to Hadoop on all Hadoop distributions. In addition, this release has dedicated SAS/ACCESS products to support Impala on Cloudera and MapR, and HAWQ on Pivotal HD distribution. Information about supported versions and Hadoop distributions can be found at <http://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoop-distributions.html>.

Hive is continuously being improved and MapReduce is not the only execution back end anymore. So with most distributions there are multiple options to consider. Ultimately you will have to compare yourself using your own data and your own system to decide what works best for your usage pattern.

Accessing data, SQL pass-through, and SAS In-Database procedures are supported for all the above mentioned SQL engines. The LIBNAME statement might have some additional options to take advantage of specific options of the SQL engine, but the rest of your SAS code can be used without change when wanting to use a different SQL engine. The SAS/ACCESS documentation has separate chapters for the above mentioned SQL engines with more details.

TIP #4: AVOID USING SAS TEMPORARY TABLES

When working with Hadoop data using SAS code, it is often common to store temporary tables in the SAS WORK library. The problem with this is that the SAS WORK library is located on the SAS server that is running the SAS code. Hadoop tables can be 10s to 100s of gigabytes or terabytes. Using WORK to store temporary data means that data will be traveling back and forth between Hadoop and SAS.

The following example demonstrates the problem code:

```
proc sql;
  connect to HADOOP
  (DATABASE=pocdb SERVER=pocserver AUTHDOMAIN="pocAuth");
  create table work.W2TUXC2M as /* avoid the use of work */
  select gld_cus_id
  from connection to HADOOP
  (SELECT * FROM sas_hive_write); /* avoid this */
  disconnect from HADOOP;
quit;
```

Instead, recode using explicit SQL pass-through and the INSERT INTO statement:


```
proc sql;
  connect to HADOOP
  (DATABASE=pocdb SERVER=pocserver AUTHDOMAIN="pocAuth");
  execute (
    INSERT INTO TABLE hiveschema2 temptable
    SELECT gld_cus_id
    FROM hiveschema1.sas_hive_write
  ) by HADOOP;
```

```
disconnect from HADOOP;  
quit;
```

TIP #5: AVOID USING PROC APPEND

PROC APPEND has a similar issue. Often PROC APPEND is generated to append one Hadoop HIVE table to another. When working with Hadoop HIVE tables, the data will travel back through the SAS server in order to perform the append. Instead of PROC APPEND, use explicit SQL to append data together in Hadoop, so that the data does not go back to SAS. You can use the OVERWRITE option to replace data if preferred when using explicit SQL, as shown below:

```
proc sql;  
  connect to HADOOP  
  (DATABASE=pocdb SERVER=pocserver AUTHDOMAIN="pocAuth");  
  execute (  
    INSERT OVERWRITE TABLE hiveschema2.temptable  
    SELECT gld_cus_id  
    FROM hiveschema1.sas_hive_write  
  ) by HADOOP;  
  disconnect from HADOOP;  
quit;
```

Note that SAS® Data Integration Studio will generate the above explicit SQL pass-through code using Hive syntax for you when using the join node. You simply have to select the explicit pass-through option on the options panel. SAS Data Integration Studio will also validate and display when your code has been correctly configured to completely push down into Hadoop. Figure 5 indicates that the join and the tables are all going to be processed in Hadoop indicated by the  symbol.

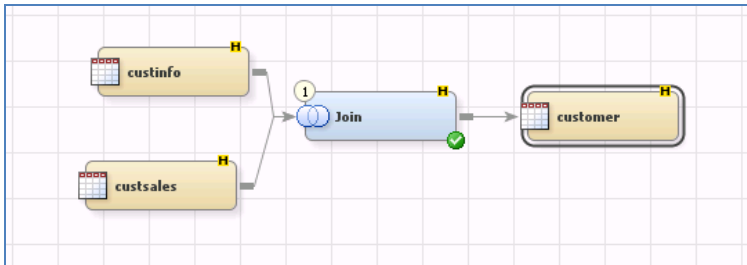


Figure 5: SAS Data Integration Studio - Job Using Explicit Pass-Through for Join

Figure 6 is an example of the explicit pass-through code that will be generated by SAS Data Integration Studio for the join transform when working with Hadoop data.

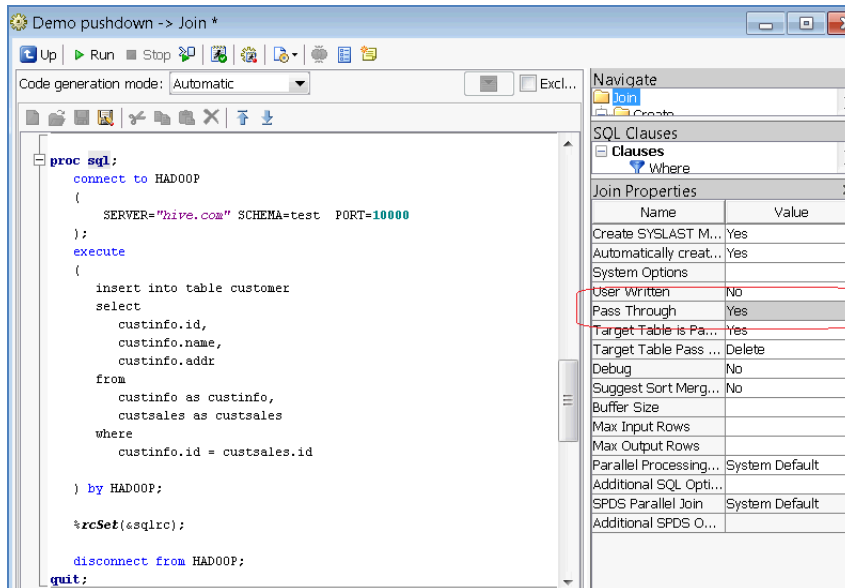


Figure 6: SAS Data Integration Studio - Generated Code

TIP #6: USING SPARK RUN TIME TO IMPROVE PERFORMANCE

SAS Data Loader for Hadoop and the SAS® Data Management Accelerator for Spark supports Apache Spark™ in Hadoop. Spark differs from traditional Hadoop MapReduce run-time processing, in that it holds intermediate results in memory rather than writing them to disk. This is very useful when you need to work on the same data set multiple times, or perform a sequence of row-based operations on a single data set. In addition, when using Spark, data will only be written to disk at the end of the sequence of transformations, not during the intermediate steps, which means that all of the processing will occur in memory. This will perform much faster than traditional Hadoop MapReduce processing, where data is landed between each operation onto disk.

In SAS Data Loader for Hadoop, the Cleanse Data, Transform Data, and Cluster-Survive directives all support the ability to leverage Spark as their run-time environment when Spark is enabled on your cluster. You can also write user-written expressions to perform your own processing on your data in the Spark run time, using the DataFlux® Expression Engine Language (EEL). If you have used DataFlux Data Management Studio (part of all SAS Data Management offerings), you might be familiar with EEL code.

SAS Data Loader for Hadoop makes it easy to enable Spark support by simply selecting Spark as the preferred run-time target in the Data Loader configuration panel as shown in figure 7. SAS Data Loader for Hadoop will fall back to using traditional MapReduce for you if your Spark engine becomes unavailable.

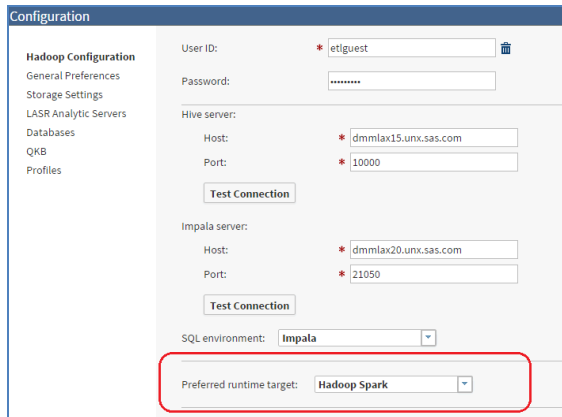


Figure 7: SAS Data Loader for Hadoop - Configuration Using Hadoop Spark as Run-Time Environment

There are a few nuances to be aware of when using Spark and the SAS Data Management Accelerator for Spark to process your data.

1. When using Spark, the maximum string length of a column that has a length attribute associated with its data type (VAR and VARCHAR in a Hive 14 based cluster) is controlled by the configuration option EXPRESS_MAX_STRING_LENGTH, which is set in the app.cfg file on each node of your cluster and defaults to 5 MB. If you need string lengths larger than this, then you need to edit this file on each of your nodes to change the default. For string type columns that are using a data type that does not have a length attribute (such as a STRING type column), maximum string length is determined by the lesser value of the configuration option EXPRESS_MAX_STRING_LENGTH or the field Maximum length for SAS columns in the Data Loader configuration panel. Note that the value of EXPRESS_MAX_STRING_LENGTH also specifies the maximum amount of memory that is allocated. For this reason, administrators should be judicious when changing the default value.
2. Spark interacts with Hive tables directly, so no modification of your data needs to be made when working with Spark. However, Hive views cannot be used as source tables to Spark. Also, the Parquet data format cannot be selected for target tables. Parquet tables can be used as source tables to Spark jobs.
3. SAS supports Spark versions 1.3.X and Spark 1.5.2 and above.

TIP #7: USE SAS IN-DATABASE PROCEDURES FOR PERFORMANCE

When using conventional processing to access data inside a data source, SAS asks the SAS/ACCESS engine for all rows of the table being processed. The SAS/ACCESS engine generates an SQL SELECT * statement that is passed to the data source. That SELECT statement fetches all the rows in the table, and the SAS/ACCESS engine returns them to SAS. Hadoop tables can be 10s to 100s of gigabytes or terabytes, which means that a lot of data will be traveling back and forth between Hadoop and SAS.

SAS® In-Database Technologies for Hadoop enables some Base SAS procedures to perform processing inside Hadoop, also known as the SAS in-database procedures. These procedures are enabled for processing data inside Hadoop, and will only return a much smaller result set for the remaining analysis that is required to produce the final output, which will result in a significant performance improvement.

The following are some of the procedures that are supported to run in Hadoop: **FREQ**, **REPORT**, **SUMMARY**, **MEANS**, **TABULATE**, **TRANSPOSE**, **MERGE**.

By default these procedures will try to take advantage of in-database processing. The **SQLGENERATION** option, which can be specified in a SAS **OPTIONS** statement and in a **LIBNAME** statement, controls whether in-database processing should be tried or not.

It is important to note when using the in-database procedures that some code can prevent in-database processing, preventing the procedure from running in Hadoop. Here are a few coding problem areas to be aware of when working with your in-database procedures:

- Limit the use of user-defined formats or make sure that you have published the formats to the database.
- Avoid the use of OBS and FIRSTOBS because Hadoop has no inherent order for the rows.
- Avoid the use of Hadoop invalid column names in your source or target.
- Some data set related options such as DBNULL, DBTYPE, will prevent pushdown.
- Function references where there is not a similar function available in Hadoop will prevent pushdown; this is particularly common when using date and time type functions.
- Use of SAS encryption options or passwords on your data sets will prevent pushdown.

TIP #8: ABOUT HADOOP FILE FORMATS

By default SAS will generate text files when writing to Hadoop. Over time SAS has introduced support for multiple file types in Hadoop and SAS 9.4M3 now supports RCFfile, ORC, Parquet. When using SAS against Hive, Avro is supported as well. Each file type has pros and cons, and as usual there are many factors that determine what file type is best for your specific usage scenario, including your usage pattern, your Hadoop vendor and Hadoop version, and your data. We will not go into details of the various file types, there is a lot of information available on the Internet about this, but we will mention some things to consider when choosing a file type, and then show you how you can specify your file type of your target table using SAS.

All of the file types mentioned take advantage of compression, therefore reducing disk foot print. However, keep in mind that compression comes with a cost, especially when writing data. Both ORC and Parquet store data in columnar format, which theoretically should provide an advantage when reading only a subset of columns of a wide table and/or a subset of the rows. For usage patterns where you write the data once and then read it multiple times, like for reporting or analytics, this performance benefit on read might outweigh the cost of the one time write. But as always your mileage will vary, so you will have to test your own usage patterns using your own data.

You can use the DBCREATE_TABLE_OPTS option to specify the file type for your output table. This option is available both in a SAS LIBNAME statement and as a SAS data set option.

To have all your output tables in a specific SAS Hadoop library stored using the ORC file format:

```
libname mydata HADOOP ... DBCREATE_TABLE_OPTS="stored as ORC";
```

To have a specific table stored using the ORC file format:

```
data mydata.table_out (DBCREATE_TABLE_OPTS="stored as ORC");  
  set mydata.table_in;  
run;
```

Note that this option can also be used in combination with proc DS2.

TIP #9: HADOOP CONFIGURATION OPTIONS

Hadoop is a complex system with many settings. Hadoop vendors provide recommendations for many configuration settings based on size, in terms of data nodes, of the cluster. The documentation contains tables for different ranges of number of nodes and for each range there is a list of values for specific configuration settings.

It's important to keep in mind that these are intended as general guidelines and are not carved in stone. Especially if the size of your cluster is close to upper-limit of a size range, then we have seen in multiple client situations that increasing the values to the recommended settings for the next size up, often resolves most (if not all) stability issues and causes certain performance issues to disappear as well.

Another thing to be aware of is that when running concurrent SAS jobs each with multiple librefs to HiveServer2, each of these librefs keep connections open for the duration of the SAS job. We have seen jobs like this fail intermittently with connection errors being shown in the SAS log. This is often caused by low (default) values for options like `hive.server2.thrift.max.worker.threads`. Increasing the value of this parameter often resolves the intermittent connection issues. In this scenario Zookeeper errors might show up as well as each HiveServer2 client will have a connection to Zookeeper. The Zookeeper `maxClientCnxns` is often set to a low default value as well and when experiencing intermittent connection errors, increasing the value of this option might help.

TIP #10: KERBEROS

This will be a very short tip. SAS supports Hadoop clusters that are configured to use Kerberos authentication and we are seeing a steady increase in the percentage of Hadoop clusters that use Kerberos. The SAS documentation has been enhanced based on customer feedback to explain in more detail how SAS/ACCESS Interface to Hadoop software and SAS In-Database Technologies for Hadoop can be configured when your Hadoop cluster uses Kerberos. The chapter called “Configuring Kerberos” in the *SAS® 9.4 In-Database Products Administrator’s Guide, Seventh Edition* has been rephrased. So even if you are not using the latest maintenance release and are using SAS against a Hadoop cluster that uses Kerberos authentication, the latest documentation might be helpful.

CONCLUSION

This paper discusses a set of practical recommendations for optimizing the performance and scalability of your Hadoop system using SAS. Topics include recommendations gleaned from actual deployments from a variety of implementations and distributions. Techniques cover tips for improving performance and working with complex Hadoop technologies such as Kerberos, techniques for improving efficiency when working with data, methods to better leverage the SAS in Hadoop components, and other recommendations. With this information, you can unlock the power of SAS in your Hadoop system.

RECOMMENDED READING

- Agresta, R., 2015. “Master Data and Command Results: Combine MDM with SAS Analytics for Improved Insights.” *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings15/SAS1822-2015.pdf>.
- Ghazaleh, David. 2016. “Exploring SAS® Embedded Process Technologies on Hadoop®.” *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings16/SAS5060-2016.pdf>.
- Rausch, N., et al. 2015. “What’s New in SAS Data Management.” *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings15/SAS1390-2015.pdf>.
- Rausch, N., et al. 2016. “What’s New in SAS Data Management.” *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings16/SAS2400-2016.pdf>
- Ray, Robert. Eason, William. 2016. “Data Analysis with User-Written DS2 Packages.” *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings16/SAS6462-2016.pdf>.
- Rineer, B., 2015. “Garbage In, Gourmet Out: How to Leverage the Power of the SAS® Quality Knowledge Base”, *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings15/SAS1852-2015.pdf>.
- SAS® 9.4 Support for Hadoop website. Available at <http://support.sas.com/resources/thirdpartysupport/v94/hadoop/>.
- SAS Data Management Community. Available at https://communities.sas.com/t5/Data-Management/ct-p/data_management

- SAS Data Management Forum. Available at https://communities.sas.com/t5/SAS-Data-Management/bd-p/data_management
- SAS Institute Inc. 2015. *Base SAS® Procedures Guide, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/proc/68954/PDF/default/proc.pdf>.
- SAS Institute Inc. 2015. *SAS® 9.4 DS2 Language Reference, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/ds2ref/68052/PDF/default/ds2ref.pdf>.
- SAS Institute Inc. 2015. *SAS® 9.4 Hadoop Configuration Guide for Base SAS® and SAS/ACCESS®, Second Edition*. Cary, NC: SAS Institute Inc. Available at <https://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoopbacg.pdf>.
- SAS Institute Inc. 2015. *SAS® 9.4 In-Database Products Administrator's Guide, Seventh Edition*. Cary, NC: SAS Institute Inc. Available at <https://support.sas.com/documentation/cdl/en/indbag/69030/PDF/default/indbag.pdf>
- SAS Institute Inc. 2015. *SAS® 9.4 In-Database Products User's Guide, Sixth Edition*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/indebug/68442/PDF/default/indebug.pdf>.
- SAS Institute Inc. 2015. *SAS/ACCESS® 9.4 for Relational Databases: Reference, Seventh Edition*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/acreldb/68028/PDF/default/acreldb.pdf>.

CONTACT INFORMATION

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

Wilbram Hazejager
 100 SAS Campus Dr
 Cary, NC 27513
 SAS Institute Inc.
 Work Phone: (919) 677-8000
 Fax: (919) 677-4444
Wilbram.Hazejager@sas.com
support.sas.com

Nancy Rausch
 100 SAS Campus Dr
 Cary, NC 27513
 SAS Institute Inc.
 Work Phone: (919) 677-8000
 Fax: (919) 677-4444
Nancy.Rausch@sas.com
support.sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.