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 YARN, 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, 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 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: USING YARN QUEUES
Even if you have a large Hadoop cluster, resources are not unlimited, and all users have to share those resources. When you want regular SAS processes to have priority over long-running queries or certain other activities in your Hadoop environment, or perhaps you want to prevent your SAS processes from consuming too much of your Hadoop resources, then it’s time to look at Apache Hadoop YARN (Yet Another Resource Negotiator).

YARN is the Hadoop cluster resource management system, whatever type of processing framework you are using (MapReduce, Spark, or Tez). The benefits promised by YARN are scalability, availability, resource-optimized utilization, and multi-tenancy.

A Hadoop administrator can define so-called YARN queues, and each queue has a set of associated resource settings. These settings specify minimum and maximum values for things like memory, virtual CPU cores, and so on. The Hadoop administrator can specify so-called placement policies that specify which default queue to use for users or groups of users.

The default queue for a user is used by SAS software unless your SAS application overrules this default. If you do override the default in your SAS application, the Hadoop administrator needs to have enabled queue overwrites; otherwise, the default queue for the user is used.

For more details about Apache Hadoop YARN, see http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.
SAS/ACCESS® Interface to Hadoop, when leveraging Hive, generates MapReduce code, and SAS Embedded Process runs purely as MapReduce jobs. Therefore, it can be completely managed by YARN.

When writing your own SAS code, you can modify the contents of the mapred-site.xml that is in the directory pointed to by the SAS_HADOOP_CONFIG_PATH environment variable and specify the MapReduce queue name to use. Here is an example:

```
<property>
  <name>mapreduce.job.queuename</name>
  <value>root.SASqueue</value>
</property>
```

Once you have done this, start your SAS session. From now on, whenever SAS code generates MapReduce jobs, this specific YARN queue is used.

You can also specify the YARN queue to use on your SAS LIBNAME statement as follows:

```
libname mydata HADOOP HOST='xxxx.yyyy.com' PORT=10000 PROPERTIES="mapreduce.job.queuename=root.SASqueue";
```

See the SAS/ACCESS documentation for more details about the PROPERTIES option in the LIBNAME statement.

When using SAS® Data Loader 3.1 for Hadoop, the YARN queue to use is extracted from the mapred-site.xml Hadoop configuration file that is available on the SAS Data Loader middle tier. Whenever SAS Data Loader initiates processing in the Hadoop cluster, it specifies that queue. If no queue was defined in the Hadoop configuration file, then SAS Data Loader falls back to using queue default, which means that the (user-specific) default queue is used.

**TIP #2: WORKING WITH HADOOP DISTRIBUTED FILE SYSTEM (HDFS)**

When working with Hive data in Hadoop, the actual content of the table is stored in the Hadoop Distributed File System (HDFS) as a file or set of files. The HDFS has a permissions model for files and directories that shares some things in common with a UNIX style POSIX model, where files and directories are owned by an owner and a group, and permissions are managed through those users.

In contrast to the POSIX model, there are no setuid or setgid bits for files. Instead, the sticky bit can be set on directories, which will prevent anyone except the superuser, directory owner, or file owner from deleting or moving files within the directory.

In SAS, this might manifest as a table that cannot be dropped or replaced by any user except the owner of the table. Multiple users can see, read, and open the table, but when they attempt to replace that table as a target, they will get a run-time error. Note that most of the vendor distributions are deployed with the sticky bit set.

SAS recommends that you turn off the sticky bit on either the /tmp directory or wherever the HDFS_TMPDIR is pointing to. The reason for this is because Work libraries will try to write, update, or delete temporary files in those locations. The permissions on the directory should be like this: drwxrwxrwx. To change the permissions using the command-line interface, as the HDFS superuser (usually hdfs), run a command similar to the following:

```
hadoop fs -chmod 0777 /tmp
```

**HDFS HOME DIRECTORIES**

Each user using the Hadoop cluster must have an HDFS home directory configured on each node in the cluster. Hadoop places files in that directory for some Hive operations. Also, because SAS Data Loader uses Oozie to run some types of jobs in Hadoop, including jobs that load data using Sqoop and Spark, it stores some temporary files in that directory.
Administrators should plan to create a user home directory and Hadoop staging directory in HDFS for each user. The user home directory in HDFS is typically `/user/myuser`. The Hadoop staging directory is controlled by the setting `yarn.app.mapreduce.am.staging-dir` in `mapred-site.xml` and defaults to `/user/myuser`. Change the permissions and owner of `/user/myuser` to match the UNIX user. The user ID must have at least the following permissions:

- Read, Write, and Delete permission for files in the HDFS directory (used for Oozie jobs)
- Read, Write, and Delete permission for tables in Hive

**TIP #3: HIGH AVAILABILITY SUPPORT**

In Hadoop, the HDFS is the primary storage system and is responsible for storing and serving all data. The HDFS has long been considered a highly reliable system. However, the HDFS has always had a well-known single point of failure because it relies on a single name node to coordinate access to the file system data. Sometimes, an HDFS outage can impact users. For this reason, you might want to consider adding high availability (HA) to the HDFS name node. You do this by adding a second name node in an active/passive configuration, so that if one goes down, the other can compensate. See your distribution vendor documentation for more details about how to configure an HA name node.

SAS supports HA for the HDFS for some products starting with the third maintenance release for SAS 9.4 and certain recent Hadoop vendor distributions. Some hotfixes might be required to support HA in some cases. HA support is available for a number of products, including the following:

- Base® SAS: FILENAME Statement for Hadoop Access Method (See Limitation 1)
- Base SAS: HADOOP Procedure (See Limitation 1)
- SAS/ACCESS Interface to Hadoop (See Limitation 2)
- SAS In-Database Code Accelerator for Hadoop

The expected behavior when enabled is the following:

- Functionality and communication with Hadoop are maintained when HDFS is configured for name node HA.
- In the case of a name node failover, SAS jobs in progress roll over to using the secondary name node, data processing continues, and there is no need to manually update client configuration.

Here are some known limitations:

Limitation 1: On Linux x64, with the connection to Hadoop orchestrated via the WebHDFS RESTFUL API (option `SAS_HADOOP_RESTFUL` set to 1), the FILENAME statement and HADOOP procedure might become unresponsive if the primary name node suffers a catastrophic failure. Restart of the SAS session might be required after the failover takes place.

Limitation 2: Performance of SAS/ACCESS Interface to Hadoop query might be affected if the primary name node suffers a catastrophic failure while the query is in progress. Queries submitted after the failover takes place are not affected.

A fix is also available through SAS Technical Support for SAS Data Loader to enable Hive HA. Similar to HDFS HA, Hive HA allows you to configure an active/passive configuration for the Hive server. Normally, SAS Data Loader generates JDBC connections using the form `jdbc:hive2://<hive-server>:10000`, which will fail in HA configurations if the Hive server is down.

With the fix in place, SAS Data Loader instead generates code similar to the following JDBC connect string if Hive HA information is found in the `hive-site.xml`:

```
jdbc:hive2://<hive.zookeeper.quorum>/serviceDiscoveryMode=zookeeper;zooKeeperNamespace=<hive.server2.zookeeper.namespace>
```
This instructs the JDBC driver to dynamically find a live Hive server. LIBNAME generation is also updated for this use case. Note that the value specified for SERVER= in the LIBNAME statement is ignored when explicitly pointing to Hive HA using the URI= option.

Support for generating Hive HA connections in SAS Data Loader can be disabled by setting the following advanced property for the “Data Loader Mid-Tier Shared Services” entry in SAS® Management Console:

\[ \text{sasdm.disable.hiveserver2.discovery} \]

**TIP #4: HIVE OPTIMIZATIONS – PART I – JOINS, FETCH TASK, FILE FORMATS**

**JOINS**

When working with data, especially big data, it is best to avoid moving data around between different computing systems. Wherever possible, you want to move your data once, and then perform your joins and other transformations.

You can use PROC SQL to perform joins on your data in Hadoop. To ensure that your join keeps the data in Hadoop to perform the join, here are a few tips:

- When using SAS PROC SQL, SAS does not pass LIBNAME-referenced cross-schema joins to Hadoop. To pass a multiple-libref join to Hadoop, the schemas for each LIBNAME statement must be identical.

  If you want to perform a cross-schema join with PROC SQL, you can use the SQL pass-through facility instead, for example:

  ```sql
  proc sql;
  connect to hadoop (user="myusr1" pw="mypwd"
  server=hxpduped port=10000 schema=default);
  ```

- Another suggestion is to use SAS Data Loader for Hadoop. SAS Data Loader is a multi-lingual code generator for Hadoop, and it is designed to automatically select the best Hadoop language to generate based on the type of transformation you want to do. For joins, SAS Data Loader generates the Hive SQL syntax to perform joins in Hadoop. This can guarantee that your data does not move around because it is using the native features of Hadoop to work with data. Figure 1 shows an example of Hive SQL generated by SAS Data Loader.

![Figure 1. Example of Hive SQL Generated by SAS Data Loader](image)

- Avoid the use of user-defined formats.
- Avoid the use of Hadoop invalid column names in your source or target.
- Some data set-related options such as DBNULL and DBTYPE will prevent pushdown.
• Function references where there is not a similar function available in Hadoop prevents pushdown; this is particularly common when using date and time type functions.

**FETCH TASK**

When working with smaller data sets in Hive, one option to speed up query performance is to consider using the fetch task. This option directly queries the data to give the result, rather than starting a MapReduce job for the incoming query. For simple queries like `select *`, it can be very fast, typically seconds instead of minutes because Hive returns the results by performing an HDFS get operation.

This feature is enabled via the following Hive configuration option:

```
hive.fetch.task.conversion = minimal;
```

In Hive 14, the fetch task is normally on by default in most distributions. When this Hive property is set, the Hive engine uses the action only for specific queries where it makes sense, like `select *`, and generates MapReduce jobs for other type of queries, when fetch would not be efficient. Statistics have to be enabled in the cluster for these settings to apply. For example, use the following code:

```
set hive.stats.autogather=true;
set hive.stats.dbclass=fs;
```

The fetch task works only on certain types of queries. The query has to be from a single data source without subqueries, aggregations, or views. It does not apply to joins, only queries. There is also a threshold value that determines the maximum size of the table that can be used with this technique. Above this threshold, Hive uses MapReduce instead. This is controlled via another setting:

```
hive.fetch.task.conversion.threshold = 1000000000;
```

In most distributions, the default is set to 1GB.

The Hive options can be set:

• at the Hadoop cluster level at the Hive server configuration level
• at the SAS level in the `hive-site.xml` connection file
• at the LIBNAME level with the PROPERTIES option; for example:

```
PROPERTIES="hive.fetch.task.conversion=minimal;
hive.fetch.task.conversion.threshold=-1";
```

For example, when a simple PROC SQL is submitted, without the property, two MapReduce jobs run, as shown in Figure 2.

**Figure 2. Example of Job Status in Hadoop**

When a simple PROC SQL is submitted with the fetch task enabled and the correct type of data query, only the MapReduce job corresponding to the actual SQL query runs, as shown in Figure 3.

**Figure 3. Example of a MapReduce Job Status**

Because this is a Hadoop setting, this feature is transparent to SAS. This means that once it is set in your cluster, SAS just takes advantage of the feature. You don’t have to write any special code to use it.
FILE FORMATS

Hadoop has various formats that can be used to store data in the HDFS. Each file type has pros and cons, and there are many factors that determine what file type is best for your specific usage scenario, such as your usage pattern, your Hadoop vendor and Hadoop version, and your data. Here is a list of the Hadoop native file formats supported by SAS:

- Delimited: This is the default type, which depends on the hive.default.fileformat configuration property
- SequenceFile
- RCFile: Available in Hive 0.6.0 and later
- ORC: Available in Hive 0.11.0 and later
- Parquet: Available in Hive 0.13.0 and later
- Avro: Available in Hive 0.14.0 and later

By default, SAS generates Delimited files when writing to Hadoop. Over time, SAS has introduced support for other file types in Hadoop including RCFile, ORC, and Parque. When using SAS with Hive, Avro is supported as well.

You can use the DBCREATE_TABLE_OPTS option to specify the file type for your output table. This option is available in both a SAS LIBNAME statement and as a SAS data set option. For example, to have all your output tables in a specific SAS Hadoop library stored using the ORC file format, use the following statement:

```
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;
```

Some of the file types mentioned above take advantage of compression, which can reduce disk space. However, keep in mind that compression comes with a cost, especially when writing data. Both ORC and Parquet store data in columnar format, which can provide an advantage when reading only a subset of columns of a wide table 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 for reading might outweigh the slower performance for writing.

Avro stores its schema as part of its metadata, which allows you to read the file differently from how you write the file. This is useful in cases where you are doing a lot of data transmissions. However, we have found during our testing that when copying a large data set into Avro format in Hive, a number of large temporary files get created in Hive. For example, an 8.6 GB table resulted in around 25 GB of persistent storage in Hive and 45 GB of temporary files. This is something to consider when moving data into and out of Avro format.

To quantify the performance advantages of the various file formats in Hadoop, we performed a series of simple performance tests. Note that this was a simple test, and your results might vary from our findings. The test environment we used was built using three servers and was performed as single user tests with no other workload. The data tables and environment were not tuned in any special way; tuning would more likely improve performance. We performed LOAD, READ, filtered READ, and JOIN tests using Hive/Parquet, Hive/SequenceFile, Hive/Avro, and Hive/ORC.

Figure 4 shows our findings using a 50 GB table:
Figure 4. Performance Characteristics of Hadoop File Types

For larger tables, the ORC and Parquet formats provided the best overall performance for the data sizes and tests we performed. These two formats also have a compression rate of about a factor of two. Avro tables increase in size as table size increases, and SequenceFile format stays about the same (for example, no compression).

As always, your mileage and usage patterns might vary, so you will have to test your own usage patterns using your own data to see what performs best for you.

TIP #5: HIVE OPTIMIZATIONS – PART II – SAS® DATA INTEGRATION STUDIO RECOMMENDATIONS

SAS Data Integration Studio supports integration with Hadoop in a variety of ways. You can create your own Hive and PIG transforms, and you can run your own MapReduce jobs. In the fourth maintenance release for 9.4, you can now run SAS Data Loader jobs from SAS Data Integration Studio. Figure 5 shows the various Hadoop transformations available:

Figure 5. Hadoop Transformation available in SAS Data Integration Studio

A new transformation allows you to select which saved SAS Data Loader jobs you want to run. One advantage of this new feature is to support integrated impact analysis. You can now see impact analysis across both SAS and Hadoop environments. Figure 6 and Figure 7 illustrate some of these new features.
One tip when working with impact analysis: if you want to trace column level impact, go into the mapping tab on the SAS Data Loader transform and create the mappings to represent how data flows into and out of the system. This is illustrated in Figure 8:

Several enhancements have been made to better ensure that the code generated in SAS Data Integration Studio pushes down to the Hadoop database. One useful setting allows you to disable the generation of column formats. This is important because some formats cannot be expressed in the Hadoop execution environment, and data would transfer unnecessarily between the two environments. By setting this option in SAS Data Integration Studio, you can avoid this extra data transfer. Figure 9 illustrates how to set this option on new and existing jobs.
One debugging tip is to set MSGLEVEL=I in your code and look at the log. SAS documents where the code is run. Figure 10 is an example of the log output:

```plaintext
27 if UnitReliability > 0.85 then score=0; Else score=1;
28 run;
INFO: Could not run DATA Step in HADOOP.
```

Another enhancement is available via SAS Technical Support when working with the Hive transformations in SAS Data Integration Studio. Figure 11 is an example of the Hive transformation:

An option has been added to the transform to allow you to turn on or off the DROP and CREATE statement, and the generated code has been modified to generate INSERT OVERWRITE instead of CREATE TABLE syntax. A partial list of the new code that gets generated is shown below:

```sql
INSERT OVERWRITE TABLE myschema.branch_dimension_new 
/* Insert Unchanged Rows from Application Layer */
SELECT app.BRANCH_SK, app.BRANCH_TYPE_DESC, app.BRANCH_NUMBER,
    app.BRANCH_NAME, app.STREET_STATE_CODE, app.STREET_POSTAL_CODE,
    app.LOAD_DTTM, app.LAST_UPD_DTTM
FROM myschema.branch_dimension_app
... other statements
```

Several changes have been made to the SAS Data Integration Studio SQL Join transform. One change now correctly handles generation of Hive SQL syntax when joining tables from multiple schemas to prefix the table name with the schema name. This is shown in the code below:

```sql
proc sql;
    connect to HADOOP
        (SERVER=pocserver PORT=10000);
    execute
    (
        INSERT INTO TABLE myschema.branch_dimension_updates
        SELECT
        ... other statements...
```
The join has also been updated to generate the pass through INSERT OVERWRITE for replace and INSERT INTO syntax for append when working with Hive data.

TIP #6: HIVE OPTIMIZATIONS – PART III – PARTITIONING

Data partitioning is useful to help improve the performances of queries. Hive supports partitioned data through the use of HCatalog (also referred to as HCat). This is a table management layer that exposes Hive metadata to other Hadoop applications.

Partitioning data in Hive can improve query performance, and it is recommended for low-cardinality variables. For example, you could partition a table of sales data by month, where each month is stored as a separate partition in Hadoop. Then, if you query with a WHERE clause based on the month, Hadoop will read only the data in the partition associated with that month.

The same option DBCREATE_TABLE_OPTS allows you to specify a PARTITION key, for example:

```sql
    proc sql;
    create table myschema2.table_part
    (DBCREATE_TABLE_OPTS="PARTITIONED BY (datebymonth date)"
      as select * from myschema1.table_unpartitioned;
    quit;
```

This code creates one file for each month in the HDFS, as shown in Figure 12:

![Figure 12. Example of HDFS Partitions](image)

TIP #7: HIVE OPTIMIZATIONS – PART IV – ENABLING HIVE STATISTICS

Statistics such as the number of rows of a table are important in Hive. One of the key uses is for query optimization. Statistics serve as the input to the cost functions of the optimizer so that it can compare different plans and choose among them.

Recent versions of Hive can store table statistics in the Hive metastore. You get this for free when a table is loaded, unless your administrator has turned table statistics off. A Hive analyze command can be run to compute table statistics at any time, as shown in Figure 13:

![Figure 13. Hive Analyze Command](image)
You might want to consider running analyze commands in batch mode to keep your statistics updated and to benefit from this pre-computed information in your queries. Alternatively, you might consider using specific formats as ORC, Parquet, Avro, and so on, which are natively storing these types of aggregates.

When statistics are available, SAS uses them to determine the fastest way to transfer data to and from Hive. For small tables, SAS avoids the MapReduce job and uses direct fetch if SAS can determine information about table size from the statistics.

**TIP #8: HIVE OPTIMIZATIONS – PART V – MAPREDUCE VERSUS TEZ**

When running complex SQL queries using Hive, potentially many different MapReduce jobs get created that cause sub-optimal performance. Hadoop vendors are taking different approaches to trying to solve this. If your Hadoop vendor supports Hive running on Apache Tez, and your cluster has been configured to support it, then data transformations generated by SAS can take advantage of it.

Tez is an extensible framework for building high-performance batch and interactive data processing applications, coordinated by YARN in Apache Hadoop. Tez improves the MapReduce paradigm by dramatically improving its speed, while maintaining MapReduce’s ability to scale to petabytes of data. When using Hortonworks Data Platform, Hive embeds Tez so that it can translate complex SQL statements into highly optimized, purpose-built data processing graphs that strike the right balance between performance, throughput, and scalability.

![Figure 14. Hive on Tez](image)

From a SAS perspective, if Tez is available in the Hadoop cluster, then it is possible to choose between a traditional MapReduce engine and Tez before submitting a SAS program that generates Hive SQL under the covers. See tip #2 in last year’s SAS Global Forum paper *Ten Tips to Unlock the Power of Hadoop with SAS®* for more information about how to get details in the SAS log about what goes on under the covers.

By default, the engine set in the Hive server configuration (hive-site.xml) is used. You can overrule this default by using the PROPERTIES option in the LIBNAME statement. It allows you to explicitly choose the engine from the SAS client. Here is an example:

```plaintext
libname mydata HADOOP HOST='xxxx.yyyy.com' PORT=10000
    PROPERTIES="hive.execution.engine=tez";
```

Customers have seen considerable performance gains when using SAS procedures that push SQL into Hive, like SQL, SUMMARY, and FREQ. See the SAS In-Database Technologies for Hadoop documentation for more details about which Base SAS procedures support processing inside Hadoop. But, as always, your mileage might vary, so you will have to test your own usage patterns using your own data.
TIP #9: SUPPORT FOR MULTIPLE SASAPP IN SAS DATA LOADER

When SAS Data Loader 3.1 was introduced in the fourth maintenance release for 9.4, the architecture changed from a single user vApp (virtual application) deployment to a centralized deployment using SAS® Intelligence Platform. This means that infrastructure components used by SAS Data Loader are defined in SAS metadata and managed using SAS Management Console. As SAS Data Loader interacts with many different parts of the Hadoop ecosystem, either directly from the SAS Data Loader middle tier or indirectly via a SAS® Workspace Server, you now see definitions for various Hadoop components in SAS metadata. This could include Hive server, Impala server, or Oozie server.

Out of the box, SAS Data Loader 3.1 supports one SAS Workspace Server, and the workspace server to use is defined at deployment time. When working in a large organization where SAS projects are strictly separated, where each project has its own SAS Workspace Server and associated SAS LIBNAME definitions, and SAS metadata access rights are defined such that a user can only work with the SAS Workspace Servers and SAS LIBNAME definitions if they are member of the project, you might need to have multiple SAS Data Loader deployments.

SAS has enhanced the functionality of SAS Data Loader 3.1 to support multiple SAS Workspace Servers using a single deployment of SAS Data Loader. All of the SAS Workspace Servers to be used by SAS Data Loader need to be configured using the exact same set of Hadoop client JARS and the same set of Hadoop configuration files (*-site.xml). This feature is available by contacting your SAS administrator.

A typical setup would be as follows:

For each project:

- Create a dedicated schema in Hive to store project-specific data.
- Create a dedicated SAS Application Server environment with a SAS Workspace Server and make sure the name represents the project. See SAS 9.4 Intelligence Platform – Application Server Administration Guide for more details about defining multiple application servers.

Figure 15. SAS Management Console-Server Manager–Showing Multiple SASApp Environments
Figure 15 shows that two SAS Application Server environments were deployed, SASApp and SASApp2, each representing a project.

- Using SAS Management Console
  - Create SAS LIBNAME definitions to represent source data locations and target data locations for the project.
  - Associate these SAS LIBNAME definitions with the corresponding SAS Workspace Server.
    - Note that a SAS LIBNAME definition can be associated with multiple SAS Workspace Server definitions (for example, to represent that multiple projects share a source data location).
  - Grant the users ReadMetadata access to the SAS Application Server, SAS Workspace Server, and SAS LIBNAME definitions for that project.
    - Note that users can be part of multiple projects.

In SAS Data Loader, the user can specify which of the SAS Workspace Servers to use using the Configuration menu. The list of available workspace servers contains only the servers that the user has been granted ReadMetadata access to.

![Configuration Menu](image)

**Figure 16. SAS Data Loader-Select SAS Workspace Server to Use**

Note that (Default) means: revert to out-of-the-box behavior; therefore, use the SAS Workspace Server that was configured using SAS Management Console as the default to use for all SAS Data Loader sessions.

The SAS Workspace Server selected will be used whenever you use SAS Data Loader until you change your selection in the Configuration menu again.

**Disclaimer:** The described functionality in this section is not available in a standard SAS Data Loader 3.1 deployment. You need to contact your local SAS administrator or SAS Technical Support to download a fix that provides this functionality.
TIP #10: DEPLOYMENT TIPS AND TRICKS

The fourth maintenance release for SAS 9.4 significantly improved the SAS deployment experience when deploying SAS into Hadoop and collecting files for SAS/ACCESS and SAS Data Loader. One of the main improvements was to move SAS Data Loader into the SAS® Deployment Wizard, and to consolidate the various pieces to all deploy under the umbrella of the SAS® Deployment Manager. Many hours were spent validating the deployment instructions to ensure that they were correct for the many different configurations of Hadoop distributions and SAS configurations.

This first and most important tip in this section is to refer to the updated SAS deployment guides when deploying SAS and Hadoop, as they capture many recommendations and best practices. This section is short because most of our recommendations have been captured in the deployment guides. However, here are a few additional tips:

- Deployment requires steps from both the SAS and the Hadoop administrators. Review the pre-install checklist because there are steps outlined for both roles prior to starting any installation or upgrade.
- Remember that you need to collect new configuration files if you upgrade your Hadoop installation, you install a new Hadoop parcel, package, service, or component on an existing cluster, or you make any other type of major configuration change.
- You need to regenerate the Hadoop SAS Embedded Process configuration file (/sas/ep/config/ep-config.xml) when you collect new JAR files, you install or upgrade Hive or HCat, you upgrade the JDK or JRE that is used by the Hadoop processes, or any other significant change to the Hadoop cluster. In some cases, the SAS Deployment Manager can now help with this step.
- The JAR collection process requires Hadoop services running in order to identify the JAR files to collect. Occasionally, the JAR collection process under- or over-collects based on your cluster and what is running on it. If you run into issues, particularly in SAS Data Loader, when attempting to first connect or run jobs, this could be the reason. Review the deployment guide and then contact your SAS administrator.

TIP #11: BONUS

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 the upper-limit of a size range, 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 has 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.

TIPS FOR WORKING WITH SENTRY

Here are a few tips if you want to get Hadoop Sentry working with SAS:

1. Log on to Hive via Beeline, and authenticate using an ID member of a group defined as one of the admin groups for Sentry.
2. Create a role for Sentry admins (for example, CREATE ROLE ADMINS).
3. Grant membership to the role to a group your ID is member of (for example, GRANT ROLE ADMINS TO GROUP hive).
4. Grant the admin role the ability to execute grant or revoke commands at the server level (for example, GRANT ALL ON SERVER SERVER1 TO ROLE ADMINS). Note that SERVER1 is the default name used at the system level to recognize Sentry. Because the name is configurable, if the name is different from the default, use that name instead of SERVER1.
5. Create one or more additional roles for SAS users (for example, CREATE ROLE SASUSERS).
6. Grant membership to the SAS user roles to one or more OS groups (for example, GRANT ROLE SASUSERS TO GROUP 'r&d').
7. Grant at least SELECT privilege on a database to the SAS user roles to allow for the LIBNAME statement to connect successfully (for example, GRANT ALL ON DATABASE DEFAULT TO ROLE SASUSERS).
8. For the SAS/ACCESS engine to successfully create and load tables, grant Read and Write access to the temporary folder to the roles created for the SAS users:

/* without HA */
GRANT ALL ON URI 'hdfs://<name node>:8020/tmp' TO ROLE SASUSERS;
/* with HA */
GRANT ALL ON URI 'hdfs://nameservice1:8020/tmp' TO ROLE SASUSERS;

If a folder different from /tmp is being used, that folder needs to be specified in place of the default one. Same for the HDFS default port number. With HA, nameservice1 is the default name used by the system to identify the name node. Should a site use a name different from the default, that name must be used when running the grant.
9. For the SAS Embedded Process to successfully work with Sentry, grant Read and Write access to the temporary folder to the roles created for the SAS users:

/* without HA */
GRANT ALL ON URI 'file:///tmp' to ROLE SASUSERS;
/* with HA */
GRANT ALL ON URI 'hdfs:///tmp' to ROLE SASUSERS;

10. To make sure SAS Embedded Process can successfully load Hive tables, do the following:

If the default temporary folder is being used, create an OS group called supergroup on the name node, and connect the user hive to it. Define the user first in case the ID doesn’t exist on the server. For example, for Linux:
groupadd supergroup
useradd/usermod -G supergroup hive

If a folder different from /tmp is being used to store temporary files, give ownership to the folder to the hive user, either at the user or at the group level.

Also, make sure that the permission bits are set to 777 and that the sticky bit is not set.

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 YARN, 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.
ACKNOWLEDGMENTS
The authors wish to thank Alex Arey, Blake Russo, Chris Watson, Clark Bradley, Mauro Cazzari, and Raphaël Poumarede for their collective insights.

RECOMMENDED READING

- SAS® 9.4 Support for Hadoop website, available at http://support.sas.com/resources/thirdpartysupport/v94/hadoop/

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