Paper 3626-2019

SAS/ACCESS® Interface to Hadoop: Experiences and Best Practices at a Large Financial Institution

Tom Keefer, D4t4 Solutions

ABSTRACT

SAS/ACCESS® Interface to Hadoop is a critical component of integrating SAS® environments to the increasing amounts of data available in Apache Hadoop. Although SAS® Viya® has enhanced features for data lift to memory, there are many day-to-day tasks that will continue to leverage SAS®9 technologies for analytics. Therefore, the reliance on an optimal integration to Hadoop with SAS/ACCESS is critical, especially for larger data and user volumes. This paper highlights best practices and experiences over and above the standard SAS documentation by sharing real-life experiences from a large financial institution and close work with SAS Research and Development. This paper focuses on using Apache Hive and the Hadoop Distributed File System (HDFS). Other SAS and open source technologies are mentioned when and if they are better choices to meet performance or business requirements.

The author is a long-time SAS and financial institution architect and administrator with extensive experience in SAS and Hadoop performance tuning and hardware.

INTRODUCTION

It's important to note that SAS 9 technologies are still an important part of many IT software portfolios despite the introduction and adoption of newer technologies such as SAS Viya®. Even though there are lots of different documentation notes covering setup and configuration for SAS ACCESS to Hadoop, there is a big hole around best practices related to strategies, use cases, debugging methods and performance improvements. This paper is mainly focused on Cloudera due to the fact many large financial institutions use it. However, all tips and methodologies should be applicable to MapR, Hortonworks and vanilla Apache Hadoop. The goal of this paper is to provide admin and power users with tips and tricks to help augment their SAS ACCESS to Hadoop experience.

It is important to have some level of understanding of Hadoop and SAS ACCESS Engine components before reading this document.

SETUP & USAGE CONSIDERATIONS

There is no sense in re-sharing the technical detail found in installation and configuration guides for SAS ACCESS to Hadoop. However, it is important to consider several topics you won't find in a typical installation guide or user guide. In this section we will cover a few of those.

UNDERSTAND HOW IT WORKS, AND HOW FAST IT CAN REALLY GO

SAS ACCESS works pretty much like any other ACCESS Engine technology from SAS. In this case SAS uses java to communicate with the HDFS and/or HIVE APIs in Hadoop. When setting up SAS to talk to Hadoop you must provide jar (java archive) and Hadoop XML configuration files from the Hadoop cluster to enable SAS as a Hadoop client. Details regarding this setup can be found in the resource links provided at the bottom of this paper. HDFS is the underlying file system in Hadoop and is spread across many machines. SAS ACCESS can and does talk to all these machines, but it does **not** do that in parallel. Read and Write speeds (not including the read query complexity) from SAS via the Java based communication mechanism to or from Hadoop can be slow. If you need to read or write data faster than the single threaded mechanism can sustain, you will need to look at other options. Strategies on this are provided later in the document.

HIVE IS NOT A SUBSTITUTE FOR AN RDBMS

It is a very common mistake to think Hive will replace your extremely optimized SQL engine found in a typical RDBMS. Hive has its place in attacking very large and un-organized data. However, it is no substitute for a SQL engine that has the benefit of many years of maturity and the advantage of many highly skilled people doing SQL optimization. There have been improvements to Hive (i.e. edition of SPARK and other optimizations). However, if you really need to run traditional SQL as fast as possible for Online Transaction processing and building analytic tables from traditional data, use the right tool for the job. These other options are tools like an RDBMS or newer technologies like Snowflake and maybe even Hadoop enhancing products such as Impala.

It has been a common mistake to dump data from an RDBMS into Hive and expect the same results and easy code migration. If you don't take the time to carefully port data, size Hadoop properly, deal with format issues, modify code, tune table and data structures, and optimize Hadoop/Hive for certain queries, you will fail. It is easy to ignore the true migration cost of these actions only to later realize you spent more in the conversion than just leaving the data in the RDBMS. Yes, Hadoop has its strengths and advantages over traditional RDBMS, but be honest with the true cost of Hive and its capabilities. It's easy to get caught in the new open source trends, just be sure to evaluate all the costs and set your expectations properly.

SENTRY AND KERBEROS

In a financial institution, you won't find many Hadoop clusters without enhanced security turned on. Cloudera uses Sentry to control role-based authorization to data and metadata stored in Hadoop. There are a few important settings related to file permissions inside HDFS for SAS to operate properly. In a Sentry enabled environment pay close attention to permissions and ownerships for temporary space inside HDFS for SAS ACCESS or you will have access issues. Review the SAS documentation in the resources section for explicit directions.

Kerberos is the typical "gate keeper" for access to Hadoop clusters in financial institutions. Proper setup is critical for users outside the SAS environment before any attempt at connecting to Hadoop is made with SAS. See the Diagnosing Problems section below for strategies to help determine where security problems are.

FIREWALL FUN

Network configuration and firewalls can be a common setup hurdle when trying to connect SAS To Hadoop. Be sure to get a list of required ports (i.e. Hive port 10000, data node 50010) and make sure they are open between the two systems. There are good lists provided by SAS Technical Support and the documentation on which ports need to be available. Firewall issues are very common setup issues. Don't forget authentication/access services like LDAP and Kerberos also have port requirements. Sometimes firewall restrictions to those services can cause issues. Keep your network engineer's instant message address handy and make sure you know their favorite beverage or snack.

A common issue for a financial customer was the addition of new data nodes into the Hadoop cluster after the initial setup and the configuration of SAS and Hadoop. Because of the timing, firewall rules for the new Hadoop data nodes were not implemented and the customer was having random job failures of data query jobs from SAS. It was discovered that the failures only occurred when data was needed by SAS from the newly added data nodes. When SAS tried to talk to these new nodes, the job would fail unless the data was replicated to another non-firewalled node. Because the query occasionally worked, it was a difficult situation to diagnose.

STRING AND OTHER VARIABLE FORMATS

SAS 9 doesn't know what to do with a STRING value when it's reading from Hadoop. This can be horribly inefficient as it causes the SAS result table to have a Character variable/column 32K in size. In a recent example, a financial customer had over 100 STRING variables in their PARQUET table stored in Hadoop. The resulting table was many times larger than the source and caused issues on the Hadoop side (size and processing of the temporary table created for extraction in YARN) and on the SAS side (size of the table). To avoid this, you can add extra metadata to the Hive table to guide SAS to a more acceptable column width. This is still not optimal as it leads to heavily padded variables (extra spaces), but significantly reduces the extra "blank" space overhead. Future versions of SAS address this issue by introducing the STRING value, but unfortunately it has not been introduced in SAS 9 only SAS Viya.

There are other formats in Hadoop which might need to be converted. Date/times occasionally have translation or formatting issues. It's important to carefully examine the date formats in Hadoop and ensure you are properly translating them to an equivalent format in SAS for processing. Although this is painful, this is typical for many data migrations when moving between different storage formats or storage systems.

MAKE FRIENDS

As with any larger financial Information Technology (IT) organization, the networking, security and Hadoop teams are all critical contacts you will need to work with as you integrate SAS into the Hadoop ecosystem. It is very important to make friends with these folks as you will need their help and access to make the changes you need to integrate effectively. As mentioned earlier, find out their favorite snacks and beverages.

READ/WRITE AND GENERAL PERFORMANCE

Reading and Writing to Hadoop is no different than reading and writing from a Relational Data store with SAS. Typically, the mantra "Minimize, then Analyze" is still very viable when working with data in Hadoop. The more processing you can do inside Hadoop before extracting data can significantly enhance performance. However, if you need to repetitively transfer data out to SAS, it might be worth creating a small DataMart in SAS to reduce constant pulling from Hadoop. You can also push more processing into Hadoop itself (Hive, SQL pushdown, etc.) or look at other mechanisms (see OTHER TECHNIQUES section below) to drive more processing into Hadoop and reduce how much data you bring back to SAS. In the meantime, here are some helpful hints and thoughts to improve your overall performance.

PROFILING, WHERE IS MY BOTTLENECK?

SAS ACCESS to Hadoop is not a fast mechanism to extract or write data into SAS. This is because it can only do single threaded reads or writes. The runtime of a query in Hive varies depending on the query complexity and volume of data its processing. You can always tweak the logic of your query to improve things. However, the resulting extraction back to SAS is highly dependent on the size of that result set and is usually takes more time than the query itself (especially with large analytic tables). The larger the result set, the longer the SAS PROC SQL runtime. This transfer back to SAS can be very lengthy and it is important to plan the time for it or try to avoid it. Therefore, the motto to remember is, it's more efficient to do all the work possible in Hadoop **before** pulling data/results back to the SAS server – "Minimize in Hadoop then Analyze in SAS". As mentioned earlier, if possible use the push down technologies of certain PROCS and other techniques to execute logic in Hadoop versus back in SAS when you can.

Below is a simple example of a query to show you how to profile your SAS job. In this example, an SQL query is run from SAS against a 200 million row Hive table with a 10 million table result set landed back in SAS. In this case, the return of data is only 2 minutes of the 5-minute runtime. However, with larger datasets, the time spent pulling data back will be much worse. Plan for this or avoid it as mentioned earlier.

Code Example

```
libname lhive Hadoop server="mycoolserver.bigbank.com"
schema=myschema port=10000 SUBPROTOCOL=Hive2;
libname fdisk "/data/junk";
proc sql;
        create table fdisk.filebackinsas as
        select * from lHive.file1 x, lHive.file2 as y
        where x.custid = y.custid2;
quit;
```

SAS Log

```
Summary Statistics for HADOOP are:
Total row fetch seconds were:                          0.003535
Total SQL execution seconds were:                      0.129411
Total SQL prepare seconds were:                      187.653832
Total SQL describe seconds were:                       0.003568
Total seconds used by the HADOOP ACCESS engine were  321.311573

NOTE: PROCEDURE SQL used (Total process time):
      real time           5:22.03
      user cpu time       2.49 seconds
      system cpu time     3.64 seconds

Data Transfer Time back to SAS for small result:   ~134 seconds
```

PARALLEL CODING

If you need to read/write data quickly from Hadoop, you will typically need to roll up your sleeves and write custom SAS code, or use a different tool (more on that later). Reading data in parallel with SAS ACCESS to Hadoop can easily be done if your data is properly partitioned. However, you will have to have a good understanding of the data to do this. As an alternative, tools such as Data Integration Studio provide a drag and drop interface to aid in this step and write code for you. For parallel reads and writes using SAS, you will need to leverage SAS CONNECT and the rsubmit feature. At a high level, you launch a series of a parallel tasks to read or write data based on a range of values in a variable or set of variables. To write data, you can drop files into a folder in HDFS and use an optional command to register the data (in Hive) and/or merge the partitioned files you have loaded. For reading, you will need to merge the data once it has landed back in SAS or ensure your next processing step can handle reading multiple data partitions stored in different SAS7bdat formats/files.

There are other options with SAS such as Scalable Performance Data Server (SPDS). Although this is not discussed in this paper it could be a viable target file format when extracting a partitioned dataset out of SAS as it allows for quickly merging/snapping results back together once it's extracted to SAS. For larger projects, it is worth looking options like SPDS and Data Integration Studio because they can drastically reduce long runtimes. If you are going to go with customer code, it's a good idea to give your customers/users a template to help them get started with parallel code. The author's company (D4t4 Solutions) has provided a lot of consulting in this area.

QUERY OPTIMIZATIONS AND PASSTHROUGH TO HADOOP

SAS has put a lot of resources in R&D to ensure both SAS Procedures like MEANS, FREQ, TABULATE as well as SQL queries are pushed down into Hadoop for processing. It is important to pay close attention to the SAS logs to monitor how well things are being processed and how many passes are being made through the data inside Hadoop. Restructuring your queries to be more optimal is a typical exercise in Relational Datastores. However, Hadoop does not have advanced Query estimators like an RDBMS. This means optimizing queries will rely on best practices for HiveQL (SQL for Hive). There are a good number of blogs and papers with suggestions on query optimization. Many of these use special Hive settings which may be added to a SAS libname or inside a PROC SQL query.

Example libname code with option to use SPARK in the Hive execution:

```
libname hdp hadoop

READ_METHOD=HDFS

schema=myschmea

HDFS_TEMPDIR="/data/myproject/work/hive/project_work"

server="mycool.hadoopserver.com"

uri="jdbc:hive2://mycool.hiveserver.com:10000/myschema;principal=hive/m
ycool.hiverserver.com@MYDOMAIN;ssl=true?hive.execution.engine=spark";
```

IMPLICIT OR EXPLICIT SQL WITH SAS AND HADOOP

It is also important to note that with SQL, there is both implicit and explicit SQL. Implicit is where PROC SQL parses through the SQL command and works in concert with the RDBMS or Hive to optimize the query. This can cause some level of rewrite to execute some things in Hive and/or some things back in SAS. This can have significant performance issues as things not translatable to Hadoop are then forced to run back on SAS and this can cause a significant amount of data to be accidently brought back to SAS. However, Implicit SQL in some cases can be very beneficial as it can help overcome things like functions in standard SQL that won't run in FedSQL (Hadoop SQL). SAS does its best to translate the SQL to be something more palatable for Hive. This is very nice early on when you just need the code to work until you have time to optimize it.

Explicit SQL basically tells SAS to not modify the code and push everything down to Hadoop to run. This is okay as long as all of the code the user writes in the SQL statement is fully supported in FedSQL. If the code is not 100% FedSQL compliant, it will fail.

It is recommended for new SAS ACCESS to Hadoop users to use implicit SQL unless it causes issues with performance.

DIAGNOSING ISSUES

When there are problems with SAS ACCESS to Hadoop, it is important to do initial tests using Hadoop client tools outside of SAS before continuing any further diagnostics inside SAS. SAS leverages Hadoop technologies like Hive and HDFS to connect to Hadoop to access data. If a user cannot execute a command in the low-level Hadoop component, it doesn’t make sense to test inside of SAS until the issue for the user is resolved.

ELIMINATE COMPLEXITY WHEN TESTING

Another reason to start testing using Hadoop client tools like Hive and HDFS on the Hadoop edge (client node) is to eliminate lower level issues first (i.e. Hive service unavailable). Keep it simple by starting at the lowest level and then add layers (i.e. firewalls, network hops, etc.). After initial tests work on the Hadoop edge node, you can move back to the SAS server to test. Testing as close to the Hadoop cluster at first and moving up the layers can eliminate a lot of wasted time and frustration.

Sample diagnostic steps when you have SAS connection or query issues:

1. Do Hive/beeline and HDFS work on the Hadoop Edge Nodes with your query?
    a. If not, fix Hadoop / client tools
    b. Only move to next step once this is fixed!
2. If possible, does SAS run on the Edge Node properly with your query?
    a. If not, check SAS configuration and client jars
    b. Only move to next step once this is fixed!
3. Run SAS back on the original SAS server as a user
    a. If not working, could be network or config

LOGGING IN SAS

Turn on every log option available in your code. These are typically more verbose logging options and are especially useful when trying to get more details out of SAS ACCESS to Hadoop when using PROC SQL and other SAS code talking to Hadoop.

```
options SASTRACE=",,,ds" sastraceloc=saslog nostsuffix;

options source source2 mprint fullstimer notes fmterr;

options msglevel=i;
```

With the above options you get detailed info on how long and where SQL execution/processing happens in Hive while using PROC SQL. This can be useful when you are optimizing execution and ensuring your code is passing through and executing more optimally inside Hadoop/Hive versus back in the SAS engine. It is not optimal when data is pulled back in its entirety to SAS for processing (as mentioned earlier). It’s more efficient when all the possible work can occur in Hadoop before pulling data/results back to the SAS server (Minimize then Analyze – as mentioned earlier).

OTHER TECHIQUES

Although these options are outside the scope of SAS ACCESS to Hadoop, they are options that could help when working with data in Hadoop.

SAS EMBEDDED PROCESS

SAS created a framework called the Embedded Process (EP) to install inside of Hadoop. It runs inside the YARN framework and can be used to parallel read, process or write data that is inside of Hadoop. It was designed to work mainly with SAS's older in-memory technologies (Lasr / High Performance Procedures / Visual Analytics). However, it is now supported with the newer in-memory analytics system, SAS Viya®. The EP is used to access (read/write/process) of tables stored in Hive or HDFS (file formats: CSV, Parquet, ORC, Control-A delimited, etc.). Since it runs inside of YARN, it cooperates with other applications running in the Hadoop cluster and also reduces the time it takes to get data to the analytics in-memory.

SAS VIYA® AND HDAT

HDAT is a SAS binary format that can be used on disk, S3 or HDFS for high speed mapping into and out of memory. It is mentioned here as it is another optional (although proprietary) way to store data in Hadoop for use with SAS's Viya product. It is extremely efficient when working with very large data volumes when SAS in-memory analytics are running alongside Hadoop. Again, this is for SAS Viya or SAS High Performance Analytics only.

SAS SPARK READER

A new capability added recently to SAS is the ability to read data created by SPARK. SPARK is an in-memory open source software tool that is typically installed along with Hadoop on the cluster and controlled by YARN. SAS can now access the data created by SPARK and read it into its in-memory analytic application SAS Viya. This is a newer capability for SAS, please contact your favorite SAS contact for more details.

CONCLUSION

Hadoop can be a very powerful tool in the financial world for helping prepare and analyze large amounts of disparate data. However, if you don't remember anything from this paper, just remember that making friends with your network and Hadoop administrator(s) are critical in the success of integrating SAS with Hadoop. Most importantly be honest with yourself on the true capabilities and costs of converting to SAS ACCESS to Hadoop for your desired tasks versus using other data storage types. Hopefully these tips will help you better plan for a successful environment where SAS and Hadoop co-exist nicely for your customers.

REFERENCES

"SAS® 9.4 Guides, Papers and Documentation for Hadoop" SAS. 2018. Available at https://support.sas.com/en/documentation/third-party-software-reference/9-4/guides-papers-for-Hadoop.html.

"SAS® 9.4 Hadoop Configuration Guide for Base SAS® and SAS/ACCESS®, Fifth Edition." SAS. 2018. Available at https://documentation.sas.com/api/docsets/Hadoopbacg/9.4_01/content/Hadoopbacg.pdf.

ACKNOWLEDGMENTS

Special thanks to Brian, Darren, Jenine, Lou and Selvakumar for contributing and their ongoing support on Hadoop projects at our favorite and other large financial institutions.

CONTACT INFORMATION

Your comments, questions and favorite IPA suggestions are valued and encouraged. Contact the author at:

Tom Keefer
D4t4 Solutions, Inc.
tom.keefer@d4t4solutions.com