Paper SAS3960-2016

An Insider's Guide to SAS/ACCESS® Interface to Impala

Jeff Bailey, SAS Institute Inc.

ABSTRACT

Impala is an open-source SQL engine designed to bring real-time, concurrent, ad hoc query capability to Hadoop. SAS/ACCESS® Interface to Impala allows SAS® to take advantage of this exciting technology. This presentation uses examples to show you how to increase your program's performance and troubleshoot problems. We will discuss how Impala fits into the Hadoop ecosystem and how it differs from Hive. Learn the differences between the Hadoop and Impala SAS/ACCESS engines.

INTRODUCTION

If you have spent any time working with Hive on Hadoop, you know performance can be a problem. This combination was designed for batch processing. Batch processing means data management and huge queries that manipulate large volumes of data. Unfortunately, business style queries were an afterthought. The problem, in a word, is high-latency. It takes Hive a while to start working, so the millisecond response times you experience on your database systems are out of reach when you are using Hive. Fortunately, there is a low-latency alternative to Hive – Apache Impala.

SAS/ACCESS Interface to Impala is an exciting product. Based on Open Database Connectivity (ODBC), it gives SAS users the opportunity to access our Hadoop and Hive data in a highly concurrent, low-latency manner.

This paper covers these topics:

1. The differences between using SAS/ACCESS® Interface to ODBC and SAS/ACCESS Interface to Impala.
2. How to configure your SAS environment so that you can take advantage of SAS/ACCESS Interface to Impala.
3. How you can discover what the SAS/ACCESS product is actually doing.
4. How to effectively move data into your Hadoop environment using this product.
5. Performance tuning your SAS and Impala environment.

This paper uses an example-driven approach to explore these topics. The examples use ODBC drivers from Progress DataDirect and Cloudera (driver created by Simba Technologies). Using the examples provided, you can apply what you learn to your own environment.

WHAT IS IMPALA?

Impala an open-source analytic, massively parallel database for Apache Hadoop. It began as an internal project at Cloudera and is currently an Apache Incubator project. You can think of it as the business intelligence (BI) SQL engine for Hadoop.

What makes it so suitable for BI? There are two reasons. First, unlike Hive (also known as HiveServer2), Impala is not written in Java. Impala is written in C++ (with some assembler thrown in) and runs as a daemon. Using Hive, you submit your query then wait while a Java virtual machine starts. In contrast, the Impala process is a daemon. It is constantly running and waiting. Second, Impala is designed to handle many users. This is referred to as high-concurrency. Hive is designed for batch processing. Work continues to close these processing gaps, but it isn't finished.

Both Hive and Impala use the same dialect of SQL. It is called HiveQL. They also use same metadata infrastructure. This is nice because it means you can run Hive and Impala on a single cluster.

Impala is currently shipped with the Cloudera and MapR distributions of Hadoop.

A GENTLE INTRODUCTION TO ODBC

Open Database Connectivity (ODBC) is a standard that was created by Microsoft. (ODBC 1.0 was released in September 1992.) ODBC has a very ambitious goal: to enable users to easily access data from any relational database using a common interface. It is intended to be the industry standard for universal data access. ODBC is very versatile, and it is the technology chosen for many SAS/ACCESS engines – including SAS/ACCESS Interface to Impala.

THE ODBC.INI FILE AND WHAT IT MEANS TO YOU

SAS/ACCESS Interface to Impala shields you from the complexities of the odbc.ini file. In fact, it greatly simplifies connecting to Impala. You do not need to learn complicated connection syntax, and you do not need to create ODBC data source names (DSN). This is not the case if you are using SAS/ACCESS Interface to ODBC. However, you might need to override one of the driver's default settings.

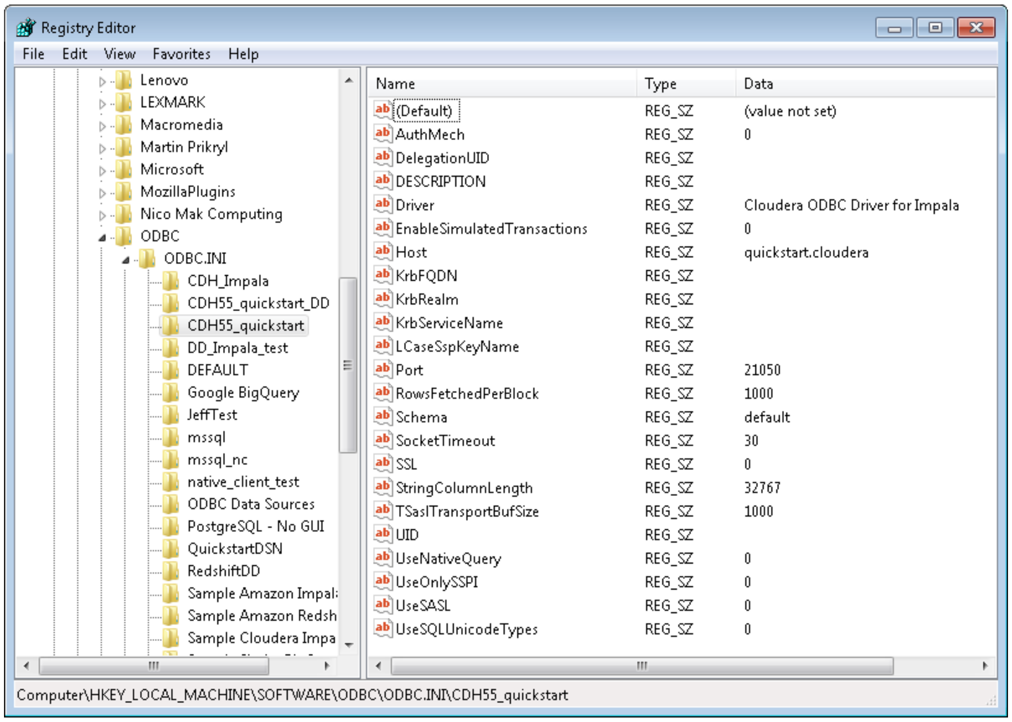
In UNIX, there is a real odbc.ini file. It is a simple text file that you can edit. Here is an example for the Cloudera Impala ODBC driver.

```
Driver=/opt/cloudera/impalaodbc/lib/64/libclouderaimpalaodbc64.so
HOST=quickstart.cloudera
PORT=21050
Database=impala
AuthMech=0
KrbFQDN=
KrbRealm=
KrbServiceName=
UID=cloudera
TSaslTransportBufSize=1000
RowsFetchedPerBlock=1000
SocketTimeout=0
StringColumnLength=32767
UseNativeQuery=0
```

Here is an example for the DataDirect Impala ODBC driver.

```
Driver=/opt/datadirect/impala7.10/lib/ddimpala27.so
Description=DataDirect 7.1 Impala Wire Protocol
ArraySize=1024
Database=impala
DefaultLongDataBuffLen=1024
DefaultOrderByLimit=-1
EnableDescribeParam=0
HostName=quickstart.cloudera
LoginTimeout=30
LogonID=cloudera
MaxVarcharSize=2147483647
Password=cloudera
PortNumber=21050
RemoveColumnQualifiers=0
StringDescribeType=-9
TransactionMode=0
UseCurrentSchema=0
```

The odbc.ini information is stored in the registry in Windows. Display 1 shows the entry for the Cloudera provided ODBC driver.


Registry Editor

File Edit View Favorites Help

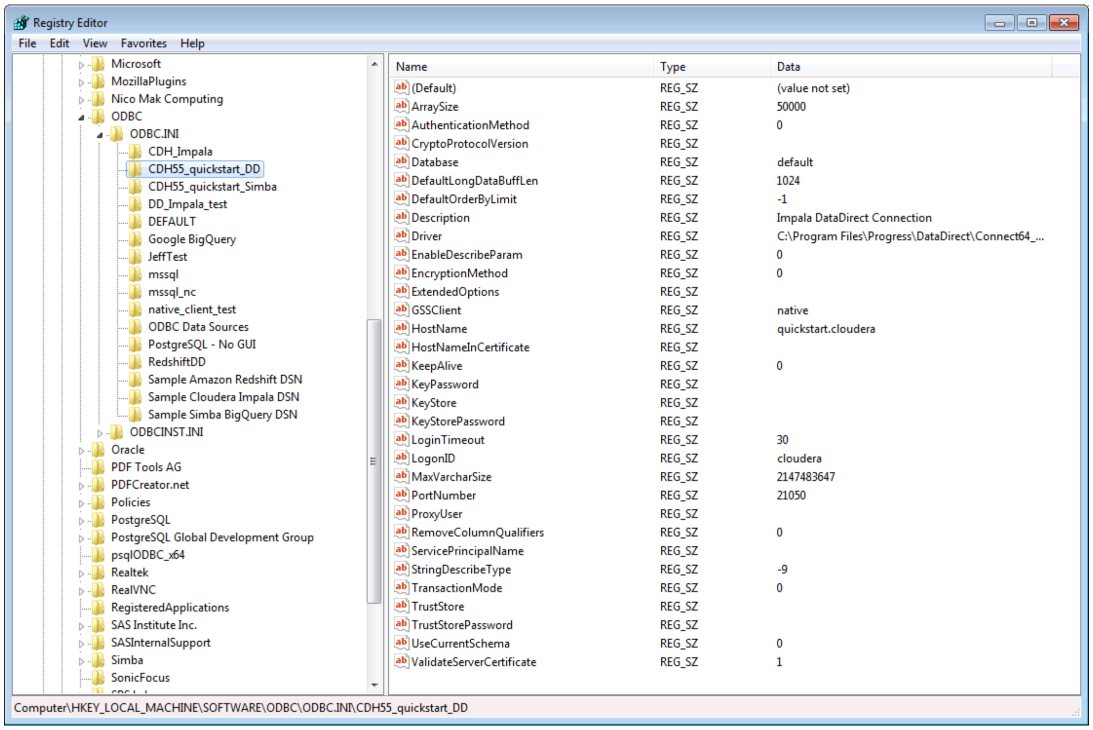
```
Lenovo
LEXMARK
Macromedia
Martin Prikryl
Microsoft
MozillaPlugins
Nico Mak Computing
ODBC
  ODBC.INI
    CDH_Impala
    CDH55_quickstart_DD
    CDH55_quickstart
    DD_Impala_test
    DEFAULT
    Google BigQuery
    JeffTest
    mssql
    mssql_nc
    native_client_test
    ODBC Data Sources
    PostgreSQL - No GUI
    QuickstartDSN
    RedshiftDD
    Sample Amazon Impal:
    Sample Amazon Redsh
    Sample Cloudera Impa
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
AuthMech	REG_SZ	0
DelegationUID	REG_SZ	
DESCRIPTION	REG_SZ	
Driver	REG_SZ	Cloudera ODBC Driver for Impala
EnableSimulatedTransactions	REG_SZ	0
Host	REG_SZ	quickstart.cloudera
KrbFQDN	REG_SZ	
KrbRealm	REG_SZ	
KrbServiceName	REG_SZ	
LCaseSspKeyName	REG_SZ	
Port	REG_SZ	21050
RowsFetchedPerBlock	REG_SZ	1000
Schema	REG_SZ	default
SocketTimeout	REG_SZ	30
SSL	REG_SZ	0
StringColumnLength	REG_SZ	32767
TSaslTransportBufSize	REG_SZ	1000
UID	REG_SZ	
UseNativeQuery	REG_SZ	0
UseOnlySSPI	REG_SZ	0
UseSASL	REG_SZ	0
UseSQLUnicodeTypes	REG_SZ	0

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\CDH55_quickstart

Display 1. Windows Registry Entry for Cloudera Impala ODBC Driver Registry Entry

Display 2 shows an entry for the DataDirect driver. Notice that the two drivers have different parameters.


Registry Editor

File Edit View Favorites Help

```
Microsoft
MozillaPlugins
Nico Mak Computing
ODBC
  ODBC.INI
    CDH_Impala
    CDH55_quickstart_DD
    CDH55_quickstart_Simba
    DD_Impala_test
    DEFAULT
    Google BigQuery
    JeffTest
    mssql
    mssql_nc
    native_client_test
    ODBC Data Sources
    PostgreSQL - No GUI
    RedshiftDD
    Sample Amazon Redshift DSN
    Sample Cloudera Impala DSN
    Sample Simba BigQuery DSN
  ODBCINST.INI
Oracle
PDF Tools AG
PDFCreator.net
Policies
PostgreSQL
PostgreSQL Global Development Group
psqlODBC_x64
Realtek
RealVNC
RegisteredApplications
SAS Institute Inc.
SASInternalSupport
Simba
SonicFocus
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
ArraySize	REG_SZ	50000
AuthenticationMethod	REG_SZ	0
CryptoProtocolVersion	REG_SZ	
Database	REG_SZ	default
DefaultLongDataBuffLen	REG_SZ	1024
DefaultOrderByLimit	REG_SZ	-1
Description	REG_SZ	Impala DataDirect Connection
Driver	REG_SZ	C:\Program Files\Progress\DataDirect\Connect64_...
EnableDescribeParam	REG_SZ	0
EncryptionMethod	REG_SZ	0
ExtendedOptions	REG_SZ	
GSSClient	REG_SZ	native
HostName	REG_SZ	quickstart.cloudera
HostNameInCertificate	REG_SZ	
KeepAlive	REG_SZ	0
KeyPassword	REG_SZ	
KeyStore	REG_SZ	
KeyStorePassword	REG_SZ	
LoginTimeout	REG_SZ	30
LogonID	REG_SZ	cloudera
MaxVarcharSize	REG_SZ	2147483647
PortNumber	REG_SZ	21050
ProxyUser	REG_SZ	
RemoveColumnQualifiers	REG_SZ	0
ServicePrincipalName	REG_SZ	
StringDescribeType	REG_SZ	-9
TransactionMode	REG_SZ	0
TrustStore	REG_SZ	
TrustStorePassword	REG_SZ	
UseCurrentSchema	REG_SZ	0
ValidateServerCertificate	REG_SZ	1

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\CDH55_quickstart_DD

Display 2. Windows Registry Entry for DataDirect Impala ODBC Driver Registry Entry

You should not consider these examples a “perfect” reference. There is little doubt that this information will have changed by the time you read this. Ideally, you would check the documentation for the driver you are using. That being said, this does provide a useful example. We will return to this example in a few moments.

SAS/ACCESS INTERFACE TO IMPALA ARCHITECTURE

ODBC-BASED ACCESS ENGINE

SAS/ACCESS Interface to Impala is an ODBC-based engine, so to use it, you must have an ODBC driver and an ODBC Driver Manager. If you are running SAS on Windows, you can use the ODBC Driver Manager that comes with the operating system.

If you are using SAS on Linux or UNIX, you can download the open-source unixODBC Driver Manager from the SAS Technical Support website or unixodbc.org. If you decide to download the unixODBC Driver Manager, make sure you get version 2.3.1 or later.

SAS/ACCESS Interface to Impala is unusual because it is not shipped with an ODBC driver. The product supports ODBC drivers from Progress DataDirect, Cloudera, and MapR.

Cloudera – You can download this ODBC driver from the Cloudera website.

MapR – You can download this ODBC driver from the MapR website.

DataDirect (Progress Software Inc.) – The DataDirect Impala ODBC driver is not free. Their drivers are licensed by many SAS customers.

Most SAS users do not need to worry about configuring ODBC drivers and managers. Installation and configuration are usually handled by systems administrators. If you want to know more about the configuration process, see *Configuration Guide for SAS 9.4 Foundation for UNIX Environments*. The details are in the “SAS/ACCESS Interface to Impala” section. There is a separate manual for Windows. (See the link in the Reference section of this paper.)

WHAT IS THE DIFFERENCE BETWEEN SAS/ACCESS INTERFACE TO IMPALA AND SAS/ACCESS INTERFACE TO ODBC?

This is one of the most common questions asked about ODBC-based SAS/ACCESS engines. It really is a great question. Another way to ask this is, “why would I pay for an ODBC-based engine when I can get SAS/ACCESS Interface to ODBC and use it with many data sources?” The answer varies depending on the SAS/ACCESS engine.

Here are the advantages to using SAS/ACCESS Interface to Impala:

Bulk Loading – As we will see shortly, SAS/ACCESS Interface to Impala supports both the WebHDFS and HDFS streaming. This functionality greatly increases the performance of moving data into Hadoop.

SAS In-Database Procedure Pushdown – SAS/ACCESS Interface to Impala converts selected SAS procedures to HiveQL, which enables Impala to do the processing work.

Create Table with the PARTITIONED BY Clause is Supported – SAS/ACCESS Interface to Impala can create partitioned tables.

Extended HiveQL Function Support – SAS/ACCESS Interface to Impala passes down more functions to the Hadoop cluster. The result is that more HiveQL is seamlessly passed to Impala, which can result in better query performance.

Extended I18N Support (internationalization) – SAS/ACCESS Interface to ODBC has support for I18N, but it requires the proper configuration of environment variable and INI file entries. This is simplified with SAS/ACCESS Interface to Impala.

USING IMPALA WITH SAS

CONFIGURATION

You probably won't be required to install and configure SAS/ACCESS Interface to Impala and the associated ODBC driver, but knowing a little about the installation and configuration can be beneficial. Here is the 2-minute version.

SAS/ACCESS Interface to Impala must be installed and available to the machine running SAS. This is the SAS server. You might hear this called the workspace server. The Impala ODBC driver must also be installed on this machine.

ODBC driver must be configured for basic processing. That sounds really simple, doesn't it? If your SAS environment is running on Windows, configuration is simple.

If your SAS environment is running on UNIX or Linux, configuration is much more involved. This paper can't cover the entire process, so here are the basic steps:

1. Install SAS and make sure it runs properly. Licensing is a common issue.
2. Install the Impala ODBC driver.
3. Configure the ODBC driver to connect to Impala. Your system administrator needs to edit the odbc.ini file.
4. Test the connection to Impala using an ODBC tool, such as isql.
5. Issue a SAS LIBNAME statement and connect to Impala. (This paper includes many examples.)

If you plan to use bulk loading (and you should) there are further configuration steps. We will discuss these steps in the Bulk Loading section of this paper.

If you are using SAS Studio (running on a Linux or UNIX environment), you might experience errors related to UTF-8. SAS Studio, unlike Base SAS, runs the UTF-8 version. Setting the following environment variable will solve this issue: EASYSOFT_UNICODE=YES.

CONNECTING TO IMPALA USING SAS/ACCESS INTERFACE TO IMPALA

There are two ways to connect to Impala using SAS/ACCESS Interface to Impala:

- LIBNAME statement
- PROC SQL CONNECT statement

LIBNAME Statement

The SAS/ACCESS LIBNAME statement enables you to assign a libref to a relational database. After you assign the libref, you can reference database objects (tables and views) as if they were SAS data sets. The database tables can be used in DATA steps and SAS procedures, such as PROC SQL, which we will discuss shortly.

Here is a basic LIBNAME statement that connects to Impala running on the Cloudera Quickstart VM:

```
libname myimp impala server="quickstart.cloudera"
                     port=21050
                     user=cloudera password=cloudera;
```

There are a number of important items to note in this LIBNAME statement:

- Libref – This LIBNAME statement creates a libref named myimp. The myimp libref is used to specify the location where SAS will find the data.
- SAS/ACCESS Engine Name – In this case, we are connecting to Impala, so we specify the IMPALA option in the LIBNAME statement.
- The SERVER= option tells SAS which Impala server to connect to. In this case, we are connecting to the Cloudera Quickstart VM. This value will generally be supplied by your system administrator.

- The PORT= option specifies which port Impala is listening on. 21050 is the default, so it is not required. It is included just in case.
- USER= and PASSWORD= are not always required. For example, the Cloudera Quickstart VM has no security enabled. That being said, some ODBC drivers require a user name and password by default. Later in this paper, you will see an example of how to override the settings of your ODBC driver.

There are times when you might need to override Impala query options. For example, you might have a query that is very resource intensive. Let's assume that your query needs additional memory in order to run quickly (or at all). Through experimentation, you have determined that 64GB is the perfect value. You can change the default value using a HiveQL SET statement. The specific option you would use is MEM_LIMIT=. Fortunately, you can do this in SAS using the DBCONINIT= LIBNAME option:

```
libname myimp impala server="quickstart.cloudera"
                     user=cloudera password=cloudera
                     dbconinit="set mem_limit=1g";
```

Note: According to the Cloudera documentation, MEM_LIMIT= is probably the most used option.

There might be times when you would like to issue multiple SET statements for a specific library. Unfortunately, specifying multiple SET statements separated by semi-colons does not work with the supported drivers.

The following example does not work:

```
/* This does NOT work */
libname myimp impala server="quickstart.cloudera"
              user=cloudera password=cloudera
              dbconinit="set mem_limit=1g;set disable_unsafe_spills=true";
```

Note: We have passed this information along to the driver vendors. Hopefully, specifying multiple SET statements using the DBCONINIT= option will work in the very near future.

What would you do if you needed to override an ODBC driver option and there is no corresponding SAS LIBNAME statement option? It seems like it would be difficult, right? Fortunately, the CONOPTS= option enables you to override ODBC driver settings.

As previously mentioned, the DataDirect driver requires both a user ID and a password by default. The Cloudera Quickstart VM does not have security enabled, so we should not have to specify a user ID and password. Fortunately, there is a way around it. The following SAS LIBNAME statement tells the DataDirect driver to not require authentication parameters:

```
libname myimp impala server="quickstart.cloudera"
                     driver_vendor=datadirect
                     conopts='AuthenticationMethod=-1';
```

There are a couple of items to notice here. First, the DRIVER_VENDOR= option is used to specify which ODBC the connection will use. Second, the AuthenticationMethod= ODBC driver option is set to -1. This setting tells the driver that the connection requires no authentication. Granted, this is not a great example but it will serve our purposes. This is an insider trick that might serve you well in the future.

You might be asking yourself, "How do I discover the available options for a specific driver?" We touched on this during our brief discussion of the odbc.ini file (and Windows registry entry). You can find these values in the odbc.ini file on UNIX or in the Windows registry. Although this paper provided a list, be sure to do some research. There could be new parameters available.

Implicit Pass-Through

SAS/ACCESS Interface to Impala generates HiveQL and sends it to Impala. Implicit pass-through references the Impala table via a LIBNAME statement. Here is an example:

```
libname myimp impala server="quickstart.cloudera"
                     user=cloudera password=cloudera;
data work.cars;
   set myimp.cars;
run;
```

You can use the SASTRACE= option to see the SQL that is being passed to the database:

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix;
```

Tip: If you use SAS/ACCESS products, you should memorize this SASTRACE statement.

This example will serve you well 99% of the time. If you would like to learn more about SASTRACE=, see *SASTRACE: Your Key to RDBMS Empowerment* by Andrew Howell. (See link in the References section of this paper.)

Joins sometimes present a problem. SAS is capable of passing implicit joins to a database if certain conditions are met. We will discuss this shortly.

Explicit Pass-Through

SAS/ACCESS Interface to Impala takes the HiveQL that you write and passes it, unchecked, to Impala. This is done via PROC SQL. If there is a CONNECT statement, explicit SQL pass-through is being used. Here is an example:

```
proc sql;
   connect to impala (server="quickstart.cloudera"
                      user=cloudera password=cloudera);
   execute(create mytable(mycol varchar(20)) by impala;
   disconnect from impala;
quit;
```

If you want to use explicit pass-through to issue an SQL SELECT statement, you must play a trick. I am including it here, just for fun:

```
proc sql;
   connect to impala (server="quickstart.cloudera"
                      user=cloudera password=cloudera);
   select * from connection to impala
      (select * from mytable);
quit;
```

Note: If the SELECT statement is embedded in an EXECUTE statement, the code will run, but nothing will be returned. This is because there is no “place” to put the results.

Explicit pass-through is very useful. For example, suppose you want to see how your Impala table is defined. You can use the HiveQL DESCRIBE FORMATTED statement:

```
proc sql;
   connect to impala (server="quickstart.cloudera"
                      user=cloudera password=cloudera);
   select * from connection to impala
      (describe formatted strcollngtest);
quit;
```

Notice the “select * from connection to” trick is required to get this to work.

JOINS

Chapter 5 of *SAS/ACCESS 9.4 for Relational Databases Reference* discusses under what circumstances join processing is passed down to the database. I highly recommend that you read this. In the meantime, let's discuss one of the most common reasons that joins are not passed to the database for processing.

Remember: In order for join processing to be eligible for push-down, the following options must be consistent between the LIBNAME statements:

- SERVER=
- PORT=
- USER=
- PASSWORD=

Notice that the SCHEMA= option is not included in the list. Can this option have different values and still produce a join that is passed to Impala? Let's experiment and see what happens. This code is simple but useful. You can use it as a test for many join situations. The following code will provide us with an answer:

```
/* create two schemas for testing */
proc sql;
   connect to impala (server="quickstart.cloudera" user=cloudera
      password=cloudera);
   execute(create schema schema1) by impala;
   execute(create schema schema2) by impala;
quit;

/* Assign two libraries with different schemas */
libname imp1 impala server="quickstart.cloudera" user=cloudera
   password=cloudera schema=schema1;
libname imp2 impala server="quickstart.cloudera" user=cloudera
   password=cloudera schema=schema2;

/* Create two simple tables */
data imp1.table1;
   x=3; output;
   x=2; output;
   x=1; output;
run;

data imp2.table2;
   x=3; y=3.3; z='three'; output;
   x=2; y=2.2; z='two'; output;
   x=1; y=1.1; z='one'; output;
   x=4; y=4.4; z='four'; output;
   x=5; y=5.5; z='five'; output;
run;

/* Display the generated SQL */
options sastrace=',,,d' sastracelog=saslog nostsuffix;

proc sql;
   select i1.x, i2.y, i2.z
     from imp1.table1 i1
        , imp2.table2 i2
    where i1.x=i2.x;
quit;
```

This code creates two Impala schemas, two SAS libraries, two tables, and joins the tables. Output 1 shows that the join was passed to Impala for processing.

```
55   proc sql;
56      select i1.x, i2.y, i2.z
57        from imp1.table1 i1
58            , imp2.table2 i2
59       where i1.x=i2.x;

IMPALA_37: Prepared: on connection 2
SELECT * FROM `schema1`.`table1`


IMPALA_38: Prepared: on connection 3
SELECT * FROM `schema2`.`table2`


IMPALA_39: Prepared: on connection 2
 select i1.`x`, i2.`y`, i2.`z` from `schema1`.`table1` i1, schema2.table2
i2 where i1.`x` = i2.`x`

NOTE: Writing HTML Body file: sashtml.htm

IMPALA_40: Executed: on connection 2
Prepared statement IMPALA_39

ACCESS ENGINE:  SQL statement was passed to the DBMS for fetching data.
60   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.78 seconds
      cpu time            0.10 seconds
```

Output 1. SASTRACE Output from a Join That Was Passed to Impala

Starting in the third maintenance release for SAS 9.4 (SAS 9.4M3), a message in the log tells you if the SQL statement was passed to the database for processing. It is a good idea to review your SAS logs and make sure that the database is doing as much processing as possible.

One of the most common questions that I am asked is, “Will this join be passed to the database?” The information detailed above enables you to answer that question easily.

SAS PROCEDURE PUSH-DOWN

SAS/ACCESS Interface to Impala supports in-database processing support for the following SAS procedures:

- FREQ
- MEANS
- RANK
- REPORT
- SORT
- SUMMARY

- TABULATE

In this example, the RANK procedure pushes processing into Impala:

```
/* LIBNAME statement must include SCHEMA= */
libname myimp impala server="quickstart.cloudera" user=cloudera
   password=cloudera schema='default';

options sastrace=',,,d' sastraceloc=saslog nostsuffix;

/* Create an Impala table */
data myimp.class;
   set sashelp.class;
run;
/* Run rank procedure in Impala */
/* LIBNAME statement must include SCHEMA= option */
proc rank data=myimp.class out=work.class_rank;
   by descending weight;
run;
```

Output 2 shows the SAS log output when the above code is run.

```
78   libname myimp impala server="quickstart.cloudera" user=cloudera
password=XXXXXXXX
78 ! schema='default';
NOTE: Libref MYIMP was successfully assigned as follows:
      Engine:        IMPALA
      Physical Name: quickstart.cloudera


IMPALA:  Called SQLTables with schema of default

IMPALA_52: Prepared: on connection 0
SELECT * FROM `default`.`class`

79   proc rank data=myimp.class out=work.class_rank;
80      by descending weight;
81   run;

 NOTE: PROCEDURE SQL used (Total process time):
       real time           0.01 seconds
       cpu time            0.01 seconds

IMPALA_53: Prepared: on connection 5
SELECT `table0`.`name`, `table0`.`sex`, `table1`.`rankalias0` AS `age`,
`table2`.`rankalias1` AS
`height`, `table0`.`weight` FROM ( SELECT `age` AS `age`, `height` AS
`height`, `name` AS `name`,
`sex` AS `sex`, `weight` AS `weight` FROM `class` ) AS `table0` LEFT JOIN (
WITH subquery0 AS
(SELECT `weight`, `age`, `tempcol0` AS `rankalias0` FROM ( SELECT `weight`,
`age`, AVG(
`tempcol1` ) OVER ( PARTITION BY `weight`, `age` ) AS `tempcol0` FROM(
SELECT `weight`, `age`,
CAST( ROW_NUMBER() OVER ( PARTITION BY `weight` ORDER BY `age` ) AS DOUBLE
) AS `tempcol1` FROM (
SELECT `age` AS `age`, `height` AS `height`, `name` AS `name`, `sex` AS
`sex`, `weight` AS
`weight` FROM `class` ) AS `subquery2` WHERE ( ( `age` IS NOT NULL ) ) ) AS
`subquery1` ) AS
`subquery0` ) SELECT DISTINCT `weight`, `age`, `rankalias0` FROM subquery0
) AS `table1` ON ( (
`table0`.`age` = `table1`.`age` ) AND ( ( `table0`.`weight` =
`table1`.`weight` ) OR (
`table0`.`weight` IS NULL AND `table1`.`weight` IS NULL ) ) ) LEFT JOIN (
WITH subquery3 AS
(SELECT `weight`, `height`, `tempcol2` AS `rankalias1` FROM ( SELECT
`weight`, `height`, AVG(
`tempcol3` ) OVER ( PARTITION BY `weight`, `height` ) AS `tempcol2` FROM(
SELECT `weight`,
`height`, CAST( ROW_NUMBER() OVER ( PARTITION BY `weight` ORDER BY `height`
) AS DOUBLE ) AS
`tempcol3` FROM ( SELECT `age` AS `age`, `height` AS `height`, `name` AS
`name`, `sex` AS `sex`,
`weight` AS `weight` FROM `class` ) AS `subquery5` WHERE ( ( `height` IS
NOT NULL ) ) ) AS
`subquery4` ) AS `subquery3` ) SELECT DISTINCT `weight`, `height`,
`rankalias1` FROM subquery3 )
AS `table2` ON ( ( `table0`.`height` = `table2`.`height` ) AND ( (
`table0`.`weight` =
`table2`.`weight` ) OR ( `table0`.`weight` IS NULL AND `table2`.`weight` IS
NULL ) ) ) ORDER BY
`table0`.`weight` DESC NULLS LAST
IMPALA_54: Executed: on connection 5
Prepared statement IMPALA_53

NOTE: SQL generation was used to perform the ranking.
NOTE: The data set WORK.CLASS_RANK has 19 observations and 5 variables.
NOTE: PROCEDURE RANK used (Total process time):
      real time           2.00 seconds
      cpu time            0.12 seconds
```

Output 2. SAS Log Output from a Running the RANK Procedure Test Code

Notice: The generated SELECT statement pushed the processing into Impala. There is also a NOTE statement, which tells us that the database performed the work.

BULK LOADING

SAS/ACCESS Interface to Impala includes the capability to rapidly load data into Hadoop distributions which are fully supported by SAS. Review the systems requirements for details. To do this, SAS/ACCESS Interface to Impala by-passes the ODBC driver and writes directly to the Hadoop distributed file system (HDFS). This is an extremely important feature. If you routinely move data from SAS to Hadoop, you should use this.

Bulk loading is special, and in order to get it to work, there is special configuration. Many SAS users do not know that PROC HADOOP and the Hadoop FILENAME statement are included in Base SAS. If your system has been configured for bulk loading, this functionality will be available to you. If your environment has been configured for PROC HADOOP and the Hadoop FILENAME statement, your bulk loading will work.

Unfortunately, it is easy to make a mistake that prevents this from working. We will not go into the details of how to configure the environment, but we will discuss it briefly.

To use bulk loading (and PROC HADOOP and the Hadoop FILENAME statement) with SAS/ACCESS Interface to Impala, you must set these environment variables:

SAS_HADOOP_JAR_PATH= SAS communicates with your Hadoop cluster via Java Archive files (JAR). These JAR files must be available to your SAS install. This environment variable points to a directory where these JAR files are saved.

SAS_HADOOP_CONFIG_PATH= In order to connect to Hadoop, SAS needs to know about your cluster. This environment variable points to a directory with the XML files that contain the information SAS needs to connect, such as server names, port numbers, and settings for many Hadoop parameters. It is important to realize that you can override many cluster settings by changing the values in these XML files.

SAS_HADOOP_RESTFUL= I really like this environment variable. Setting SAS_HADOOP_RESTFUL=1 tells SAS to use WebHDFS when it communicates with HDFS. WebHDFS is an HTTP REST API. Using WebHDFS limits the reliance on JAR files. This is a useful debugging tool. If bulk loading fails, you might want to set this environment variable to 1. If it fixes the problem, you have a JAR error.

At this point I imagine you are asking, “How do I get these JAR and XML files?” I like to point out that a normal user will not have to worry about this. Your SAS administrator will handle it for you. But what if you are the SAS administrator? Fortunately, there is a simple answer – the SAS Deployment Manager. We cannot cover this topic here. You can read all about it in the *SAS 9.4 Hadoop Configuration Guide for Base SAS and SAS/ACCESS*. (See the link in the References section of this paper.)

After your environment is ready, you will want to test it. Let’s run through a very simple example.

This code creates an Impala/Hive table and uses bulk loading to populate it with data. I am using the Cloudera Quickstart 5.5 VM. The BULKLOAD=YES data set option tells SAS to use bulk loading.

```
libname myimp impala server="quickstart.cloudera"
              user=cloudera password=cloudera;

data myimp.cars (bulkload=yes);
   set sashelp.cars;
run;
```

Here are the steps that the Impala engine will use to create the Cars table and bulk load it with data:

1. SAS issues two CREATE TABLE statements. The first one creates the Impala table. The second statement creates a temporary table.
2. SAS uploads the Cars data into a UTF-8 delimited text file that lives in the HDFS `/tmp` directory. It is common for this to fail the first time it is done (on the system). The cause: you don’t have proper permissions on the `/tmp` directory.
3. SAS issues a LOAD DATA Hive statement to move the data file from the HDFS `/tmp` directory into the temporary file.
4. SAS issues an INSERT INTO statement that inserts the data into the Impala table.
5. SAS deletes the temporary table.

Output 3 shows the SAS log from a working bulk load session. I added comments that describe the processing.

```
13   data myimp.cars (bulkload=yes);
14      set sashelp.cars;
15   run;
IMPALA_4: Prepared: on connection
```

```
SELECT * FROM `cars` WHERE 0=1

NOTE: SAS variable labels, formats, and lengths are not written to DBMS
tables.

/*CREATE IMPALA TABLE */
IMPALA_5: Executed: on connection 2
CREATE TABLE `cars` (`Make` VARCHAR(13),`Model` VARCHAR(40),`Type`
VARCHAR(8),`Origin`
VARCHAR(6),`DriveTrain` VARCHAR(5),`MSRP` double,`Invoice`
double,`EngineSize` double,`Cylinders`
double,`Horsepower` double,`MPG_City` double,`MPG_Highway` double,`Weight`
double,`Wheelbase`
double,`Length` double)


/*CREATE TEMPORARY TABLE */
IMPALA_6: Executed: on connection 2
CREATE TABLE bl_cars_1770654674(Make VARCHAR(13),Model VARCHAR(40),Type
VARCHAR(8),Origin
VARCHAR(6),DriveTrain VARCHAR(5),MSRP double,Invoice double,EngineSize
double,Cylinders
double,Horsepower double,MPG_City double,MPG_Highway double,Weight
double,Wheelbase double,Length
double) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\001' LINES TERMINATED
BY '\012' STORED AS
TEXTFILE

/* SAS MOVED DATA INTO HDFS */
NOTE: There were 428 observations read from the data set SASHELP.CARS.
NOTE: The data set MYIMP.cars has 428 observations and 15 variables.


/* SAS ISSUED LOAD DATA STATEMENT TO MOVE DATA FROM /tmp INTO TEMP TABLE */
IMPALA_7: Executed: on connection 2
LOAD DATA INPATH '/tmp/bl_cars_1770654674.dat' INTO TABLE
bl_cars_1770654674

/* COPY DATA FROM TEMP TABLE INTO IMPALA TABLE */
IMPALA_8: Executed: on connection 2
INSERT INTO `cars`
(Make,Model,Type,Origin,DriveTrain,MSRP,Invoice,EngineSize,Cylinders,Horsep
ower,MPG_City,MPG_Highw
ay,Weight,Wheelbase,Length) SELECT
Make,Model,Type,Origin,DriveTrain,MSRP,Invoice,EngineSize,Cylinders,Horsepo
wer,MPG_City,MPG_Highwa
y,Weight,Wheelbase,Length FROM bl_cars_1770654674

/* DELETE THE TEMP TABLE */
IMPALA_9: Executed: on connection 2
DROP TABLE bl_cars_1770654674

NOTE: DATA statement used (Total process time):
      real time           12.42 seconds
      cpu time            0.10 seconds
```
Output 3. Output from a SAS/ACCESS Interface to Impala Bulk Load

Output 4 shows the results of the same job without bulk loading.

```
IMPALA:  Called SQLTables with schema of NULL
20   data myimp.cars;
21      set sashelp.cars;
22   run;


IMPALA_12: Prepared: on connection 1
SELECT * FROM `cars` WHERE 0=1

NOTE: SAS variable labels, formats, and lengths are not written to DBMS
tables.

IMPALA_13: Executed: on connection 2
CREATE TABLE `cars` (`Make` VARCHAR(13),`Model` VARCHAR(40),`Type`
VARCHAR(8),`Origin`
VARCHAR(6),`DriveTrain` VARCHAR(5),`MSRP` double,`Invoice`
double,`EngineSize` double,`Cylinders`
double,`Horsepower` double,`MPG_City` double,`MPG_Highway` double,`Weight`
double,`Wheelbase`
double,`Length` double)


/* IF YOU SEE LOTS OF QUESTION MARKS, THERE WAS NO BULK LOADING */
IMPALA_14: Prepared: on connection 2
INSERT INTO `cars`
(`Make`,`Model`,`Type`,`Origin`,`DriveTrain`,`MSRP`,`Invoice`,`EngineSize`,
`Cylinders`,`Horsepower
`,`MPG_City`,`MPG_Highway`,`Weight`,`Wheelbase`,`Length`)  VALUES ( ? , ? ,
? , ? , ? , ? , ? , ?
, ? , ? , ? , ? , ? , ? , ? )

NOTE: There were 428 observations read from the data set SASHELP.CARS.

IMPALA_15: Executed: on connection 2
Prepared statement IMPALA_14

NOTE: The data set MYIMP.cars has 428 observations and 15 variables.
NOTE: DATA statement used (Total process time):
      real time           27.35 seconds
      cpu time            0.01 seconds
```

Output 4. Output from a SAS/ACCESS Interface to Impala Job That Does Not Use Bulk Loading

Remember: If you see lots of question marks [?] in your output, bulk loading was not used.

THE UNDERLYING FILE

Think of Impala (and Hive) as a database. Most of the time this works well. At other times, it will come back to haunt you. This brings us to schema-on-read. Schema-on-read is one of the concepts you need to understand. It means that the underlying data (which is stored in a file) is checked when it is read. If you use SAS bulk load to load your data, you don't have to worry about this. If you copy a file to HDFS and then issue a CREATE TABLE statement that references the file, you do need to worry about it. The file will be checked when you read it using SQL. It if doesn't match the table, the query will fail.

Hidden in the above discussion is the subtle concept about the file. In a database, you never think about a file. The database handles this for you. The Hadoop world is different. You can choose the underlying file type. Parquet is the recommended file type for Impala.

Apache Parquet is a compressed columnar storage format. It was initially developed by Cloudera and is now an Apache Software Foundation project. The reason that you want to use Parquet is performance. It is much faster than using text as the underlying file format. SAS uses the DBCREATE_TABLE_OPTS= data set option to tell Impala to store the data as a Parquet file:

```
 libname myimp impala server="quickstart.cloudera"
                      user=cloudera password=cloudera);

data myimp.cars_parquet (bulkload=yes
                          dbcreate_table_opts='stored as parquetfile');
   set sashelp.cars;
run;
```

Output 5 shows the SAS log from a working bulk load session that creates a Parquet file as the underlying file format. Notice that the target table is stored as a Parquet file, but the temporary table stored as text.

```
307  data myimp.cars_parquet (bulkload=yes dbcreate_table_opts='stored as
parquetfile');
308     set sashelp.cars;
309  run;


IMPALA_126: Prepared: on connection 4
SELECT * FROM `cars_parquet` WHERE 0=1

NOTE: SAS variable labels, formats, and lengths are not written to DBMS
tables.
IMPALA_127: Executed: on connection 12
CREATE TABLE `cars_parquet` (`Make` VARCHAR(13),`Model` VARCHAR(40),`Type`
VARCHAR(8),`Origin`
VARCHAR(6),`DriveTrain` VARCHAR(5),`MSRP` double,`Invoice`
double,`EngineSize` double,`Cylinders`
double,`Horsepower` double,`MPG_City` double,`MPG_Highway` double,`Weight`
double,`Wheelbase`
double,`Length` double) stored as parquetfile

IMPALA_128: Executed: on connection 12
CREATE TABLE bl_cars_parquet_1771256467(Make VARCHAR(13),Model
VARCHAR(40),Type VARCHAR(8),Origin
VARCHAR(6),DriveTrain VARCHAR(5),MSRP double,Invoice double,EngineSize
double,Cylinders
double,Horsepower double,MPG_City double,MPG_Highway double,Weight
double,Wheelbase double,Length
double) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\001' LINES TERMINATED
BY '\012' STORED AS
TEXTFILE

NOTE: There were 428 observations read from the data set SASHELP.CARS.
NOTE: The data set MYIMP.cars_parquet has 428 observations and 15
variables.
```

```
IMPALA_129: Executed: on connection 12
LOAD DATA INPATH '/tmp/bl_cars_parquet_1771256467.dat' INTO TABLE
bl_cars_parquet_1771256467

IMPALA_130: Executed: on connection 12
INSERT INTO `cars_parquet`
(Make,Model,Type,Origin,DriveTrain,MSRP,Invoice,EngineSize,Cylinders,Horsep
ower,MPG_City,MPG_Highw
ay,Weight,Wheelbase,Length) SELECT
Make,Model,Type,Origin,DriveTrain,MSRP,Invoice,EngineSize,Cylinders,Horsepo
wer,MPG_City,MPG_Highwa
y,Weight,Wheelbase,Length FROM bl_cars_parquet_1771256467


IMPALA_131: Executed: on connection 12
DROP TABLE bl_cars_parquet_1771256467

NOTE: DATA statement used (Total process time):
      real time           11.71 seconds
      cpu time            0.03 seconds
```

Output 5. Output from a SAS/ACCESS Interface to Impala Bulk Load Which Uses a Parquet File as the Underlying Data Store

32K STRING THING

If you are going to use SAS and Hadoop (including Impala), you need to understand the 32K String Thing. Here is the problem: the Hive/Impala STRING data type becomes a $32767. format in SAS. Think about this for a second. When you read in one character, it takes 32k in SAS. What type of problems can this cause?

Here is my favorite: a customer contacts SAS Technical Support and reports a problem with SASWORK filling up. Increasing the size of SASWORK didn't fix the problem although it did postpone it. This track bounced around until the technical support consultant asked for sample code. They quickly determined that data was being read from Hadoop. Because of this, they found someone who could explain what was happening.

The problem: When SASWORK tables are created from Hive or Impala tables, every column with a STRING data type was given a $32767. format.

There are ways to avoid this situation.

- Use the Impala VARCHAR type in your Impala tables. This type became available in Impala 2.0. This is the preferred solution.
- Use the DBMAX_TEXT= SAS LIBNAME statement and DATA set option to truncate all STRING columns in the Impala table. While this can greatly reduce the space requirements, it does not entirely stop the problem.
- Use the ODBC driver options discussed in this paper.

BIGINT

This is what the SAS/ACCESS documentation says about using the Impala BIGINT with SAS:

"Converting an Impala BIGINT to a SAS numeric can result in a loss of precision because the internal SAS eight-byte, floating-pint format accurately preserves on 15 digits of precision. An Impala BIGINT preserves up to 19 digits of precision."

What exactly does this mean? Fortunately, a little code highlights the issue.

```
proc sql;
   connect to impala (server="quickstart.cloudera"
                      user=cloudera password=cloudera);
   execute(create table BigIntTest (x bigint)) by impala;
   execute(insert into BigIntTest values (9223372036854775807)) by impala;
quit;

libname myimp impala server="quickstart.cloudera" user=cloudera
password=cloudera;

proc sql;
   select * from myimp.BigIntTest;
quit;
```

A simple PROC SQL SELECT statement returns the value 9223372036854775808. Notice: this is not the value that was inserted into the table. This is the crux of the matter. You cannot assume that SAS will get the same number that was inserted into the Impala table. When you think of using BIGINT in your calculations, you can expect to have questionable results.

IMPALA QUERY TUNING BASICS

There is no way to cover Impala performance tuning completely. The world of Impala simply moves too fast. This paper covers some basic topics and points out where you can find more information. In fact, two of the topics presented here, explain plan and query profile, are very involved. As a SAS user, you might not ever use them. That being said, it is important to know they exist if for no other reason than to mention to your Hadoop administrator if there is a problem.

HOW DO I FIND THE SQL STATEMENT THAT SAS IS SENDING TO IMPALA?

This is THE most important query tuning trick you can know. In fact, we discussed this previously, but it is so important it deserves another mention. If you ever want to get a tattoo, this would be a great candidate:

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix;
```

COMPUTE STATISTICS

Impala needs information about the data in your tables so that it can determine the best way to satisfy a query. This is where catalog statistics comes in. Catalog statistics include things like how many distinct values a specific row or index has and the distribution of the data across nodes in the cluster. Accurate statistics play a large role in your query performance. You should gather statistics on every table you create or load. It is very important that you gather these statistics after the data has been loaded. If you gather these statistics prior to loading the data, it might hurt performance.

The COMPUTE STATS command gathers statistics for your Impala tables. Here is an example:

```
proc sql;
   connect to impala (server="quickstart.cloudera"
                      user=cloudera password=cloudera);
   execute(compute stats proc.txn_fact) by impala;
quit;
```

If you are running this command from SAS, you must use explicit pass-through. For more information see the *Cloudera Impala Guide*. (See the link in the References section of this paper.)

USE EXPLAIN PLAN

It is very difficult to solve performance problems if you have no idea what Impala is actually doing with your query. The steps that Impala takes to solve your request is called an execution plan. The tool you use to see this plan is called EXPLAIN. There is no way to fully cover this feature in the paper.

The EXPLAIN command is simple; the output complicated. Here is an example:

```
proc sql;
   connect to impala (server="quickstart.cloudera"
                      user=cloudera password=cloudera);
   execute(explain select count(*) from proc.txn_fact) by impala;
quit;
```

If you are running this code from SAS, you must use explicit pass-through. For more information see the *Cloudera Impala Guide*. (See the link in the References section of this paper.)

USE QUERY PROFILE

The query profile is very involved. I mention it just so you know it exists. You can read more about on the Cloudera website.

CONCLUSION

Impala opens your Hadoop environment to your many BI users. It provides incredible performance and concurrency. SAS/ACCESS Interface to Impala brings these powerful benefits to SAS. Using SAS and Impala is not difficult, but there are many things you must remember. This paper has detailed configuration requirements, connecting with SAS LIBNAME statements, the SQL procedure, bulk loading, and performance tuning tips.

Your next step is to use this new knowledge in your everyday work. Make sure you read the SAS/ACCESS documentation. It will help you get the most from your SAS, Hadoop, and Impala investment. You will be happy you did.

REFERENCES

Bailey, Jeff. 2014. “An Insider’s Guide to SAS/ACCESS Interface to ODBC.” *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc. Available at https://support.sas.com/resources/papers/proceedings14/SAS039-2014.pdf.

Cloudera, Inc. 2016. *Cloudera Impala Guide*. Available at http://www.cloudera.com/documentation/enterprise/5-4-x/topics/impala.html.

Cloudera, Inc. 2015.“The Apache Impala Cookbook.” Available at http://www.slideshare.net/cloudera/the-impala-cookbook-42530186

Howell, Andrew. 2015. “SASTRACE: Your Key to RDBMS Empowerment.” *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings15/3269-2015.pdf

Progress Software Corporation. 2016. *DataDirect Connect Series for ODBC Reference*. Available at http://media.datadirect.com/download/docs/odbc/allodbc/help.html.

Progress Software Corporation. 2016. *DataDirect Connect Series for ODBC User’s Guide*. Available at http://media.datadirect.com/download/docs/odbc/allodbc/help.html.

SAS Institute Inc. 2016. *Configuration Guide for SAS 9.4 Foundation for UNIX Environments*. Cary, NC: SAS Institute Inc. Available at https://support.sas.com/documentation/installcenter/en/ikfdtnunxcg/66380/PDF/default/config.pdf

SAS Institute Inc. 2016. *Configuration Guide for SAS 9.4 Foundation for Microsoft Windows for x64*. Cary, NC: SAS Institute Inc. Available at https://support.sas.com/documentation/installcenter/en/ikfdtnwx6cg/66385/PDF/default/config.pdf

SAS Institute Inc. 2015. *SAS 9.4 Hadoop Configuration Guide for Base SAS and SAS/ACCESS*. Cary, NC: SAS Institute Inc. Available at https://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoopbacg.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.

Simba Technologies Inc. 2015. *Simba Cloudera Impala ODBC Driver with SQL Connector, Quickstart Guide for Windows*. Vancouver, BC Canada: Simba Technologies Inc. Available at http://cdn.simba.com/products/Impala/doc/Simba_Impala_ODBC_QuickstartGuide.pdf

ACKNOWLEDGMENTS

The author extends his heartfelt thanks and gratitude to the following individuals:

David Baccanari Jr., Progress Software Corporation

Pat Buckley, SAS Institute Inc.

Chris DeHart, SAS Institute Inc.

Marie Dexter, SAS Institute Inc.

Salman Maher, SAS Institute Inc.

Julien Mansier, Progress Software Corporation

Tom Newton, Simba Technologies Inc.

Bill Oliver, SAS Institute Inc.

RECOMMENDED READING

- *SAS® and Hadoop Technology: Overview.* Available at http://support.sas.com/documentation/cdl/en/hadoopov/68100/PDF/default/hadoopov.pdf.
- *SAS/ACCESS® 9.4 for Relational Databases: Reference, Seventh Edition.* 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 author:

Jeff Bailey
SAS Institute Inc.
Jeff.Bailey@sas.com
www.linkedin.com/in/jeffreydbailey