Global Enablement & Learning

SAS with Hadoop: Performance considerations & monitoring strategies

Date: June, 2016

Contact Information:

RAPHAEL POUMAREDE

Principal Technical Architect, Global Enablement and Learning

☎ +33 (0) 614706554

raphael.poumarede@sas.com

THE POWER TO KNOW®

Table of Contents

0 Introduction and setting expectations

This document is not an exhaustive guide on all possible optimizations, but rather a collection of tricks and best practice reminders coming from the field experience. It might help the consultant on the ground when performance issues arise in a SAS with Hadoop environment.

A very general recommendation for performances, when SAS interacts with an external data store is to avoid the download of remote data on the SAS machine and to use in-database processing instead. Using SAS in-database processing, you can run scoring models, some SAS procedures, DS2 thread programs, and formatted SQL queries inside the data source.

This same recommendation is even more important in the Hadoop world, as data stored in Hadoop can be massive. Source and target tables can be in terabytes and petabytes, and only a hundreds- or thousands-core Hadoop distributed platform will be able to crunch the data in acceptable time. It will not even be possible to bring back data of this size across internal networks.

So our main goal will be to push down most of the data processing to Hadoop. We will talk about in-Hadoop processing.

The table below is based on the Introduction to SAS In-Database Processing and presents the current available in-Hadoop features and the requirements in terms of software.

In-Hadoop feature	Software Required
Scoring models	Base SAS® SAS/ACCESS® Interface to the data source SAS® Scoring Accelerator SAS® Enterprise Miner™ SAS® Factory Miner (analytic store scoring) SAS® Scalable Performance Data Server (optional) SAS® Model Manager (optional)
Base SAS procedures: FREQ REPORT SUMMARY/MEANS TABULATE	Base SAS SAS/ACCESS Interface to the data source
Base SAS procedures: TRANSPOSE (preproduction)	Base SAS SAS/ACCESS Interface to the data source SAS® In-Database Code Accelerator (SAS® Data Loader for Hadoop)
DS2 threaded programs (across Hadoop nodes)	Base SAS SAS/ACCESS Interface to the data source SAS In-Database Code Accelerator (SAS Data Loader for Hadoop)
DATA step scoring programs	Base SAS SAS/ACCESS® Interface to Hadoop
Data quality operations	Base SAS SAS/ACCESS Interface to Hadoop SAS In-Database Code Accelerator SAS Data Loader for Hadoop
Extract and transform data	Base SAS SAS/ACCESS Interface to Hadoop SAS Data Loader for Hadoop

As SAS® software and Hadoop are constantly being developed, the reader should always check the current documentation to confirm what the supported features are as the list is likely to change over time.

The following page is maintained with current supported Hadoop versions for the various SAS products:
http://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoop-distributions.html

The SAS Embedded Process is the core component used by products such as SAS Scoring Accelerator, SAS In-Database Code Accelerator, and SAS® Data Quality Accelerator to enable SAS advanced analytics processing inside Hadoop. It will also be used for specific use cases like asymmetric HPA processing, SAS LASR Analytic Server parallel lift from Hadoop, and so on.

However, if you don't have the SAS Embedded Process in your deployment, by using the best practices and optimization techniques, you can make sure your data management operations (PROC SQL, basic procedures, or the SAS DATA step) will be successfully converted by the SAS/ACCESS engine to run **inside** Hadoop.

The first section of the document will focus on SAS/ACCESS best practices and tips. This is aimed at maximizing the data management operations to be completed by the Hadoop cluster. The Hadoop processing framework was designed to leverage distributed processing across the Hadoop nodes from the outset.

In the second section, we will assume that the SAS Embedded Process has been deployed inside the Hadoop cluster, and we will focus on the way to leverage it to run SAS analytics operations directly in-Hadoop.

In the third section, lessons learned from several field experiences and feedback are shared. Many recommendations from this document are coming from these experiences and feedback.

Finally the last section will give the main options and tools to monitor (and eventually troubleshoot) SAS analytics processing in Hadoop.

Note: This version of the document does not cover:

- SAS® High Performance Analytics
- SAS/ACCESS® to Impala
- SAS Scoring® Accelerator
- SAS® Data Quality Accelerator

1 Data Management (SAS/ACCESS)

1.1 Push data management down to Hadoop!

1.1.1 Implicit and explicit SQL pass-through

In SAS code you have two ways to interact with an external Database: **implicit** and **explicit pass-through**.

When you use a SAS LIBNAME statement in a DATA step or PROC SQL, it is **implicit SQL pass-through** and SAS will to convert your SAS code to an SQL code that the target DBMS can process. The SAS/ACCESS engine will attempt to delegate most of the processing work to the Database in order to avoid a SELECT * order.

On the other hand, **explicit SQL pass-through** is a coding syntax that allows the user to write/submit database-specific SQL that SAS will pass untouched to that database.

The SAS user, usually prefers to work with **implicit SQL pass-through** as it is easier to write, does not require skills in the specific DBMS SQL syntax, and is also generally the resulting code of GUI wizards or solutions.

Nevertheless, depending on the functions used or the way the code is written, the conversion is not always possible for the SAS/ACCESS engine, in which case the data is downloaded from the remote DBMS on the SAS server for local processing using SAS data sets.

In the real world, Hadoop clusters can be really big in terms of resources and tables sizes. Customers can have 100 to over 1,000 nodes in their Hadoop environments. In these environments due to the size of data, meaning that the source and target tables are in the hundreds of gigabytes if not terabytes, it is not possible to bring the data back to SAS to run the query efficiently.

As pointed out in the next section (Best practices), there are techniques to increase the ability of the SAS SQL planner to interpret SQL code and to allow the implicit SQL pass-through to be as efficient as explicit SQL pass-through.

1.1.2 Best practices

General guidelines when you develop and run SAS data operations with Hadoop are listed here:

- Push down the SQL processing to Hive as much as possible:

  - Avoid merging SAS data with Hive data. It is recommended that you transform the SAS data set in a Hive table and run the merge inside Hive to leverage distributed processing and avoid network traffic between SAS workspace server node and the Hadoop cluster. See Passing joins to Hadoop.
  - Avoid using a function that will bring Hadoop data back to the SAS server, See Passing SAS functions to Hadoop.
  - Use the SASTRACE option to display details of the communications between SAS and Hadoop.
  - Compare with SQL explicit pass-through.
  - Use the "magic options" (see below) to help the SQL planner to push down processing in Hadoop.
- Monitor end-to-end progress of a job to identify under-optimized queries.
- Identify the bottleneck in the chain. When a data operation from SAS is longer than expected, try to translate it in HIVEQL and to run it directly in the server. If the times are similar, Hive optimization might be required.

1.1.3 Magic options

- Use DBIDIRECTEXEC when using CREATE TABLE AS SELECT (CTAS) operations. (See Create a Hive table from SAS (CTAS).)
- Use SQL_FUNCTIONS=ALL, which allows for SAS functions that have slightly different behavior from corresponding Hadoop functions that are passed down to Hadoop. (See Passing SAS Functions to Hadoop.)

1.2 Optimize your SELECT

1.2.1 Fetch task instead of MapReduce application

Nearly every time SAS interacts with a Hive table (for example, to display the table attribute from SAS® Enterprise Guide® or SAS® Enterprise Miner™ graphical assistants), it performs SELECT * to Hive, which will run a MapReduce lob to download the full table.

Additionally, each time you use the PROC SQL with a HADOOP LIBNAME table (implicit SQL pass-through), a preliminary SELECT * order is always sent, triggering a MapReduce job.

It is not a problem if you have a small table, but as soon as you start dealing with million-row tables (not even really big data) it can take several minutes to perform the above operations.

There is a solution. You can force Hive to use a **fetch task** to perform this initial query.

With a fetch task, Hive directly goes to the file and gives the result, rather than start a MapReduce job for the incoming query. For simple queries like SELECT * with limit, it is much faster. In this case, Hive will return the results by performing an HDFS operation (hadoop fs –get equivalent).

Of course it will not be efficient for all type of queries. But when the Hive property **hive.fetch.task.conversion** is set to minimal, the Hive engine will use the fetch action only for specific light queries where it makes sense (like our automatic SELECT * to get the table's metadata) and will generate MapReduce jobs for other types of queries (where the fetch is not efficient).

Finally, there is also another parameter related to this: **hive.fetch.task.conversion.threshold**. In Hive 0.10 to Hive 0.13, the default is -1 (no limit). In Hive 0.14 and later, the default is 1G. This setting indicates that if the table size is greater than the value, it will use MapReduce rather than the fetch task to handle the query.

The SAS user experience with SAS Enterprise Guide or SAS Enterprise Miner when interacting with large Hive tables will immediately be improved when these options are in place. (See the example below.)

This Hive options can be set:
- At the Hadoop cluster level, in the Hive server configuration level
- At the SAS level, in the hive-site.xml connection file
- At the LIBNAME level with the PROPERTIES option

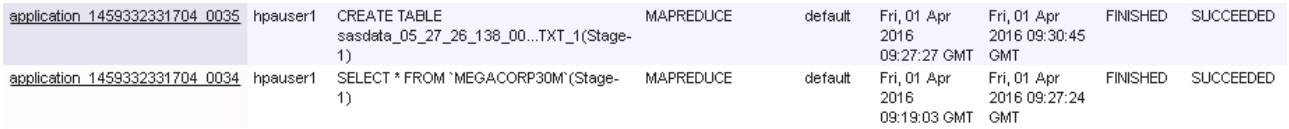
It is recommended that you set it at the SAS level to generally enhance the user experience when interacting with Hive tables in SAS clients.

1.2.2 Demonstration

In the SAS LIBNAME statement, the Hive option has to be set as below:

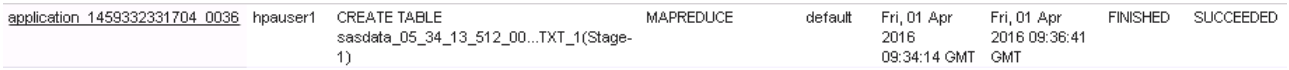
```
PROPERTIES="hive.fetch.task.conversion=minimal;hive.fetch.task.conversion.threshold=-1";
```

When a simple PROC SQL is submitted without the property, two MapReduce jobs will run (the first one being the automatic SELECT query):



application_1459332331704_0035	hpauser1	CREATE TABLE sasdata_05_27_26_138_00...TXT_1(Stage-1)	MAPREDUCE	default	Fri, 01 Apr 2016 09:27:27 GMT	Fri, 01 Apr 2016 09:30:45 GMT	FINISHED	SUCCEEDED
application_1459332331704_0034	hpauser1	SELECT * FROM `MEGACORP30M`(Stage-1)	MAPREDUCE	default	Fri, 01 Apr 2016 09:19:03 GMT	Fri, 01 Apr 2016 09:27:24 GMT	FINISHED	SUCCEEDED

When a simple PROC SQL is submitted with the property, only the MapReduce job corresponding to the actual SQL query will run. (We don't see anything for the FETCH action corresponding to the first SELECT * executed from SAS):



application_1459332331704_0036	hpauser1	CREATE TABLE sasdata_05_34_13_512_00...TXT_1(Stage-1)	MAPREDUCE	default	Fri, 01 Apr 2016 09:34:14 GMT	Fri, 01 Apr 2016 09:36:41 GMT	FINISHED	SUCCEEDED

Example:

SQL query example	
proc sql; select count(*) as nb,min(unitreliability) as minur from hivelib.MEGACORP30M; **quit;**	
Without the fetch options property	With the fetch options property
NOTE: PROCEDURE SQL used (Total process time): real time 11:49.20 cpu time 0.39 seconds	NOTE: PROCEDURE SQL used (Total process time): real time 2:32.03 cpu time 0.13 seconds

1.2.3 References

- See the SAS Note "Queries run against a large Hive table might be slow": http://support.sas.com/kb/57/776.html
- Understand hive.fetch.task.conversion and hive.fetch.task.conversion.threshold properties in Hive: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties

1.3 The “Order by” hurdle

1.3.1 Hive limitation

From the Hive documentation:

"There are some limitations in the "order by" clause…in order to impose total order of all results, there has to be one reducer to sort the final output. If the number of rows in the output is too large, the single reducer could take a very long time to finish."

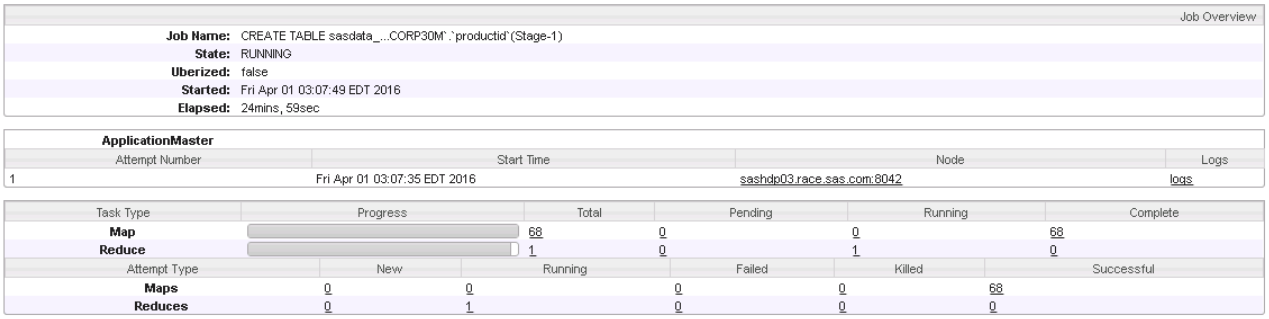
The ORDER BY clause on a large table is very costly in Hive. SAS code optimization can avoid the use of the ORDER BY statement in Hadoop.

1.3.2 Demonstration

The behavior associated with this problem is illustrated below with a DATA MERGE step with a 30-million-row table. This DATA MERGE step will trigger the ORDER BY in Hive:

```
DATA hivelib.TARGET;
     MERGE hivelib.MEGACORP30M (IN=A)
   work.DEFECT_PRODUCTS(IN=B);
BY PRODUCTID;
IF A ;
IF B THEN TARGET = 1; ELSE TARGET = 0;
RUN;
```

The mapper phase can be quite fast, but although the reducer phase will have a quick start, it can then take a lot of time (hours) to complete, with very small progressions at the end.



Job Overview

Job Name:	CREATE TABLE sasdata_...CORP30M`.`productid`(Stage-1)
State:	RUNNING
Uberized:	false
Started:	Fri Apr 01 03:07:49 EDT 2016
Elapsed:	24mins, 59sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Fri Apr 01 03:07:35 EDT 2016	sashdp03.race.sas.com:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map		68	0	0	68
Reduce		1	0	1	0

Attempt Type	New	Running	Failed	Killed	Successful
Maps	0	0	0	0	68
Reduces	0	1	0	0	0

Once completed, we can display the detail of the job execution:

Job Name:	CREATE TABLE sasdata_...CORP30M`.`productid`(Stage-1)
User Name:	hpauser1
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Apr 01 03:07:34 EDT 2016
Started:	Fri Apr 01 03:07:49 EDT 2016
Finished:	Fri Apr 01 03:34:25 EDT 2016
Elapsed:	26mins, 35sec
Diagnostics:	
Average Map Time	1mins, 2sec
Average Shuffle Time	5mins, 13sec
Average Merge Time	4sec
Average Reduce Time	19mins, 43sec

We can see that out of 26 minutes of total time to process the CTAS (CREATE TABLE AS SELECT) query, the reducer phase (ORDER BY) **with the unique reducer took almost 20 minutes**.

The bigger and wider the table is, the longer it will take.

Now if we rewrite the DATA MERGE step with a PROC SQL, then the CREATE TABLE does not contain the ORDER BY clause.

Job Overview

Job Name:	CREATE TABLE `default`.`TAR...t2.`productid`(Stage-4)
User Name:	hpauser1
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Apr 01 09:37:18 EDT 2016
Started:	Fri Apr 01 09:37:25 EDT 2016
Finished:	Fri Apr 01 09:46:29 EDT 2016
Elapsed:	9mins, 3sec
Diagnostics:	
Average Map Time	1mins, 35sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Fri Apr 01 09:37:21 EDT 2016	sashdp04.race.sas.com:8042	logs

Task Type	Total	Complete
Map	68	68
Reduce	0	0

Attempt Type	Failed	Killed	Successful
Maps	0	0	68
Reduces	0	0	0

In this case there is no reducer phase and the CTAS step only takes 9 minutes.

Example:

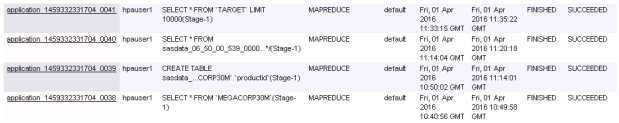
<table>
<tr><th>Bad way</th><th>Better way (optimized query)</th></tr>
<tr><td>

```
libname hivelib hadoop
        server="sashdp01.race.sas.com"
        user=&sysuserid
        password="whatever"
        database=default
        subprotocol=hive2
properties="hive.execution.engine=mr";


/*Bad way - merge local table with
hive table, triggers a ORDER BY IN
HIVE*/
DATA hivelib.TARGET;
     MERGE hivelib.MEGACORP30M (in=A)
   work.DEFECT_PRODUCTS(IN = B);
BY PRODUCTID;
IF A ;
IF B THEN TARGET = 1; ELSE TARGET = 0;
RUN;
```

</td><td>

```
/*optimize CTAS operations*/
options DBIDIRECTEXEC;
libname hivelib clear;
libname hivelib hadoop
        server="sashdp01.race.sas.com"
        user=&sysuserid
        password="whatever"
        database=default
        subprotocol=hive2
/*use fetch conversion  and strict
mode control to have an error message
when costly ORDER is generated*/
properties="hive.fetch.task.conversion
=minimal;hive.fetch.task.conversion.th
reshold=-1;";
/* upload DEFECT_PRODUCTS table in
HIVE to avoid Network Traffic*/
data hivelib.DEFECT_PRODUCTS;
set work.DEFECT_PRODUCTS;
run;
/*rewrite the query in SQL instead of
DATA MERGE to avoid the ORDER BY*/
proc sql;
create table hivelib.TARGET as
select t1.*,
     (CASE WHEn t2.PRODUCTID is not
null then 1 else 0 end) AS TARGET
        FROM hivelib.MEGACORP30M t1
                LEFT JOIN
hivelib.DEFECT_PRODUCTS t2 ON
(t1.PRODUCTID = t2.PRODUCTID);
quit;
```

</td></tr>
<tr><td>Number of Map Reduce Job : 4</td><td>Number of MapReduce job(s) : 1</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Processing time : 52 min 17 sec</td><td>Processing time : 9 min 28 sec</td></tr>
<tr><td>NOTE: DATA statement used (Total process time):
real time 52:17.06
cpu time 6:26.10</td><td>NOTE: PROCEDURE SQL used (Total process time):
real time 9:28.63
cpu time 0.41 seconds</td></tr>
</table>

1.3.3 Strict Mode

From the Hive documentation:

*In the strict mode (i.e., **hive.mapred.mode=strict**), the "order by clause" has to be followed by a "limit" clause. The limit clause is not necessary if you set hive.mapred.mode to nonstrict.*

The idea of the strict mode is prevent the client application queries from generating the ORDER BY on the full table.

You can specify it in the LIBNAME PROPERTIES, for example:

```
PROPERTIES="hive.mapred.mode=strict";
```

And if you try to run a bad query you will receive this error message:

```
ERROR: Prepare error: Error while compiling statement: FAILED:
SemanticException 1:386 In strict mode, if ORDER BY is specified,
       LIMIT must also be specified. Error encountered near token
'product_id'
```

1.4 Create a Hive table from SAS (CTAS)

Over time, implicit pass-through has become more and more powerful and supports most of the DBMS interaction scenarios. However in specific cases, running explicit SQL pass-through or helping the SAS/ACCESS engine to translate more efficiently the PROC SQL code into native operations in the database can drastically improve the performance.

Running a simple CREATE TABLE AS SELECT (CTAS) in PROC SQL is one of these cases in Hadoop.

Using the **DBIDIRECTEXEC** option will force specific operations (as table creation or delete) to be passed to the DBMS.

In the Hadoop case, the coordination of Hive, MapReduce, and HDFS operations will be optimized in many cases when this option is set for CTAS operations.

Example:

<table>
<tr><td colspan="2">SQL query example</td></tr>
<tr><td colspan="2">

```
proc sql;
create table hivelib.megacorp30mprofits
as select mdate,profit,revenue,expenses from hivelib.megacorp30m;
quit;
```

</td></tr>
<tr><td>Without option DBIDIRECTEXEC</td><td>With option DBIDIRECTEXEC</td></tr>
<tr><td>Log messages</td><td>Log messages</td></tr>
<tr><td>SQL_IP_TRACE: None of the SQL was directly passed to the DBMS.
…
LOAD DATA INPATH '/tmp/sasdata-2016-04-04-08-29-48-131-e-00005.dlv' OVERWRITE INTO TABLE `MEGACOPR30MPROFITS`</td><td>SQL_IP_TRACE: passed down query: CREATE TABLE `default`.`megacorp30mprofits` as select TXT_1.`mdate`, TXT_1.`profit`,
TXT_1.`revenue`, TXT_1.`expenses` from `default`.`MEGACORP30M` TXT_1
…
SQL_IP_TRACE: The CREATE statement was passed to the DBMS.</td></tr>
<tr><td>Processing time : 5 min 45 sec</td><td>Processing time : 3 min 28 sec</td></tr>
<tr><td>NOTE: PROCEDURE SQL used (Total process time):
real time 5:45.29
cpu time 1:19.97</td><td>NOTE: PROCEDURE SQL used (Total process time):
real time 3:28.08
cpu time 0.22 seconds</td></tr>
</table>

Note: Notice the big difference in CPU time. On multi-user platform, that difference in real-time could get even bigger.

1.5 MapReduce or Tez?

Apache Tez is a data processing framework available in Hortonworks distributions. It can be used as an alternative to MapReduce and is presented by Hortonworks as improving the MapReduce paradigm by dramatically improving its speed, while maintaining MapReduce's ability to scale to petabytes of data.

Important Hadoop ecosystem projects like Apache Hive and Apache Pig can use Apache Tez.

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 (when a SAS/ACCESS Interface to Hadoop LIBNAME statement is submitted against Hive).

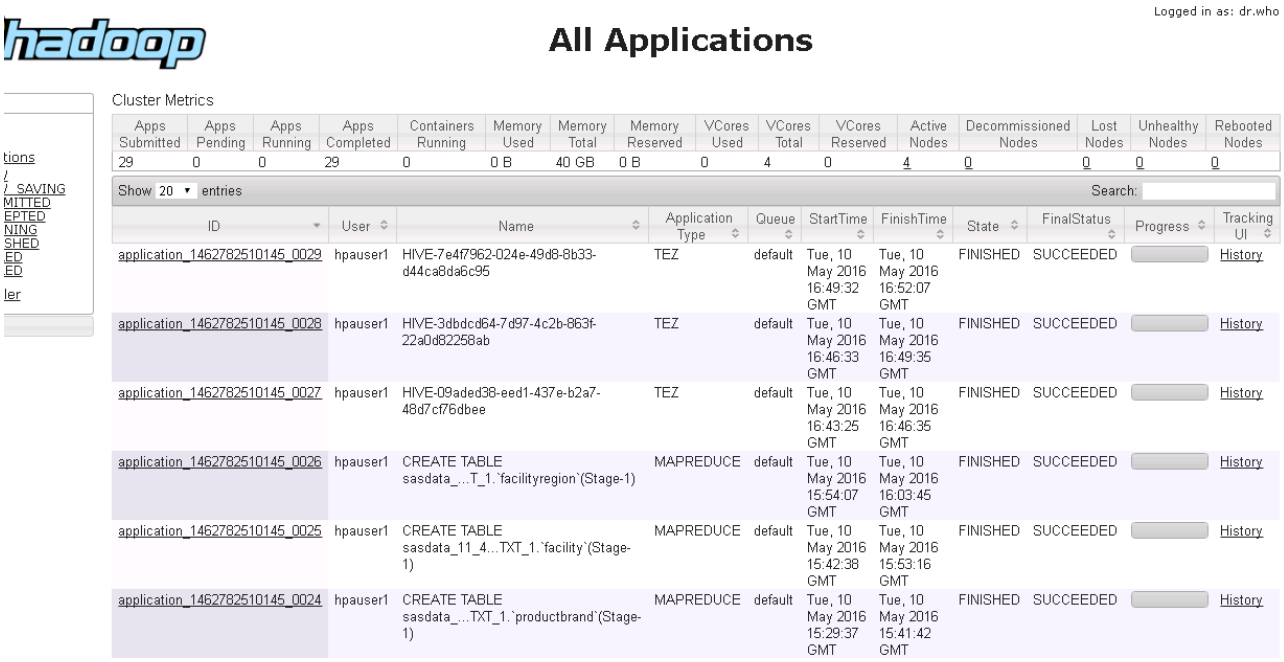
By default, the engine set in the Hive server configuration is used. Nevertheless, using the PROPERTIES option in the LIBNAME statement allows you to explicitly choose the engine from the SAS client.

```
PROPERTIES="hive.execution.engine=mr";
```

Or

```
PROPERTIES="hive.execution.engine=tez";
```

The screen capture below shows MapReduce and Tez jobs corresponding to a few tests (PROC FREQ, PROC SUMMARY, and PROC SQL) executed on a 30-million-row table:



While the RACE environment used for the test (Babar collection) is not a candidate to run official benchmarks on, these few tests did show significant differences in execution times (real and CPU) between MapReduce and Tez engines:

Test in Hive	**MapReduce** as Hive engine	**Tez** as Hive engine
PROC SQL	real time 10:30.97 cpu time 2.15 seconds	real time 2:31.38 cpu time 1.51 seconds
PROC SUMMARY	real time 11:30.04 cpu time 2.25 seconds	real time 3:00.14 cpu time 1.67 seconds
PROC FREQ	real time 13:00.68 cpu time 3.39 seconds	real time 3:08.08 cpu time 1.70 seconds

Note: However, if Tez is not the most important application in Hadoop, Tez could become a concern because it is designed to take and hold resources for this job versus going back to YARN for resources (like all other applications). As an example, a customer case has been reported where a client wanted DS2 code to have top priority over Hive queries, but Tez did not allow this to happen.

1.6 File formats and partitioning

1.6.1 File formats in Hive (ORC, Avro, Parquet, and others)

By default, when a Hive table is created from SAS, it is stored as Text. Nevertheless, other formats that bring new features (compression, partitioning, metadata, and statistics) are available in Hive and can be used by SAS.

SAS can work with the ORC file format in Hive as described in the article How to persist native SAS data sets to Hadoop (Hive) in the SAS Communities Library.

The option allowing you to create a file in an alternative format is DBCREATE_TABLE_OPTS=.

The option can be used at the LIBNAME level or in the CREATE TABLE statement.

Example:

```
libname orc hadoop
        server="sashdp01.race.sas.com"
        user=&sysuserid
        password="whatever"
        database=orctables
        subprotocol=hive2
            DBCREATE_TABLE_OPTS='STORED AS ORC'
properties="hive.fetch.task.conversion=minimal;hive.fetch.task.conversion.thresh
old=-1;hive.execution.engine=mr";
```

or:

```
proc sql;
create table seq.megacorp5M_sequence (DBCREATE_TABLE_OPTS='STORED AS
SEQUENCEFILE')
                as select * from megacorp5M;
quit;
```

Using a new popular Hadoop storage format does not necessarily mean that you will observe significant performance changes in all cases. It really depends on the use case and data profiling. Generally, this is what has been observed:

- Whole table operations will perform better with row-oriented files (Text, Sequence).
- Subset operations and applying predicates will work better with column-oriented files (Parquet, ORC, and RCFile).
- Serialization formats (Avro) are best for transmission and storage. Avro stores its schema as part of its metadata. This means that how you read the file can differ from how you write the file, making it flexible and easy to add data.

Note: During the tests, when copying a large SAS data set (30 million rows) into an Avro file in Hive, a lot of large files are temporarily created. The HDFS is almost saturated. (An 8.6 GB SAS table resulted in around 25 GB of persistent storage in Hive and 45 GB as temporary files.)

1.6.2 Partitioning

Data partitioning is often used when working with traditional database management systems to improve performances of queries. Hive supports partitioned data. When data is registered in the Hive metastore, it can be partitioned and HCatalog (also referred to as HCat) can be used to access that data through standard Hive mechanisms. (Check your Hive version and the HCat support in your Hadoop environment.)

HCatalog is a table management layer that exposes Hive metadata to other Hadoop applications. It is part of Apache Hive. It strongly supported by Hortonworks. It is also supported by Cloudera, but not as well as by Hortonworks. A REST interface is also available called WebHCat.

Partitioning data in Hive can improve query performance. It is recommended that you use partitioning on low cardinality variables used often in the queries.

See: http://blog.cloudera.com/blog/2014/08/improving-query-performance-using-partitioning-in-apache-hive/ for examples.

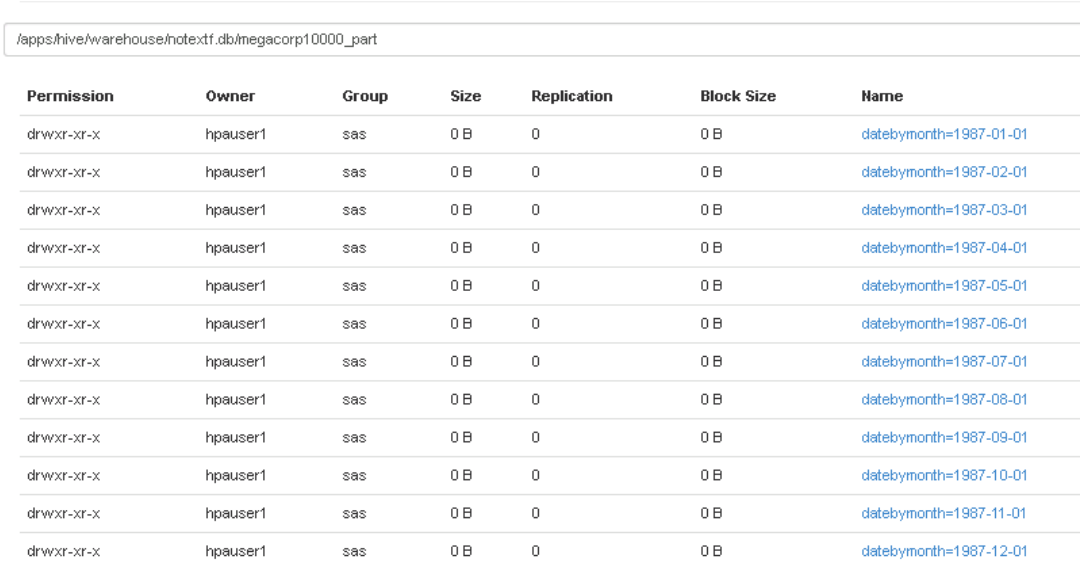
The same option, DBCREATE_TABLE_OPTS=, allows you to specify PARTITION keys.

For example:

```
proc sql;
create table notextf.megacorp10000_part(DBCREATE_TABLE_OPTS="PARTITIONED BY
(datebymonth date)")
                as select * from local.megacorp10000;

quit;
```

This code creates one file for each different month in HDFS:



/apps/hive/warehouse/notextf.db/megacorp10000_part

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-01-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-02-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-03-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-04-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-05-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-06-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-07-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-08-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-09-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-10-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-11-01
drwxr-xr-x	hpauser1	sas	0 B	0	0 B	datebymonth=1987-12-01

1.6.3 SPDE format

Included with Base SAS (does not require any SAS/ACCESS engine), the SPDE format in Hadoop has many advantages (partitioning, index, MapReduce jobs for WHERE processing, and so on) and can be faster than Hive formats in several cases. (See Hadoop file type benchmark.) Here are some of the advantages:

- Accessible by apps other than SAS from Hive, thanks to the recent SerDe format developed for it.
- Could be faster than Hive access when working with SAS :
    - Depending on the queries (no need to deal with Hive, direct access via HDFS)
    - Depending on type of loading (for example, for parallel load in SAS LASR Analytic Server)
    - When working with HPA procedures (see benchmark results in annex)
- SPDE also provide some of the traditional SAS features as :
    - Encryption
    - File compression
    - Member-level locking
    - SAS indexes
    - SAS password
    - Special missing values
    - Physical ordering of returned observations
    - User-defined formats and informats

However, most data access pulls data back to the SAS client for processing. One exception is the parallel data access using the SAS Embedded Process for Hadoop (comes with the SAS/ACCESS module). Another exception is WHERE filtering using the ACCELWHERE= LIBNAME or data set option, which allows you to return only a subset of data to the SAS client.

Several excellent papers, including best practices, are available on how to benefit from this format. See References for additional details on SPDE format in HDFS.

1.6.4 SASHDAT

SASHDAT is the SAS High Performance proprietary format whose purpose is to allow operations between data in memory and data in HDFS to be as fast as possible. It will provide the best performance for the following tasks:

- Load HDFS data in SAS LASR Analytic Server
- Save SAS LASR Analytic Server table in HDFS
- Run a High Performance procedure (PROC HPDS2, HPSUMMARY, HPLOGISTIC, HPREG, HPGENSELECT, and so on) on data stored in HDFS.

Nevertheless, SASHDAT is really designed for these in-memory solutions (SAS LASR Analytic Server procedures, and other High Performance procedures such as HPREG, HPLOGISTIC, and others, and HPDS2).

The data in SASHDAT cannot be read from standard SAS (DS SAS procedure, SQL, DS2), and it cannot be accessed from Hive. (It is mainly a memory representation in HDFS blocks).

Additional details on SASHDAT engine can be found here:
http://support.sas.com/documentation/cdl/en/inmsref/68736/HTML/default/viewer.htm#n1r9w66xkxubg7n1gta7yb6w1z25.htm

1.7 Other tricks

1.7.1 Changing the default maximum length for SAS character columns

In some cases, the message below appears in the SAS logs:

```
WARNING: SAS/ACCESS assigned these columns a length of 32767. If resulting SAS
character variables remain this length, SAS
         performance is impacted. See SAS/ACCESS documentation for details.
Columns followed by the maximum length observed were:
         order_date:10, cust_address:269
```

This message means that SAS has converted a Hive table column into a SAS character column with the default maximum length (32K). It can happen, for example, if some columns are stored with the STRING type in Hive (instead of VARCHAR(x)).

Of course, this is not optimal, as the result will be an overly wide SAS data set (imagine if multiple columns are in this situation), and also columns might not get the proper SAS formats (for example, SAS date format).

The solution to this problem is to use the SAS table properties in Hive.

In the example above we can see that the length of the cust_address STRING in Hive never exceeds 269 characters. So we can issue Hive ALTER TABLE statements to add SAS table properties to the Hive table definition:

```
ALTER TABLE orders_fact SET TBLPROPERTIES ('SASFMT:cust_address'='CHAR(300)')
ALTER TABLE orders_fact SET TBLPROPERTIES ('SASFMT:order_date'='DATE(9.0)')
```

The resulting SAS data set that is created from the Hive table has a much smaller observation width, which helps SAS save disk space and reduce CPU consumption. It also allows you to associate the proper SAS date format.

Note: A macro is offered through SAS technical support that will examine all your Hadoop tables and make appropriate format modifications.

Reference:
http://support.sas.com/documentation/cdl/en/acreldb/69039/HTML/default/viewer.htm#p1rj6miqsmhercn17lz0xatfqd4l.htm (See the section “Leverage Table Properties for Existing Hive Tables”.)

1.7.2 Using Hive Statistics

1.7.2.1 Availability of the statistics

From the Hive documentation:

Statistics such as the number of rows of a table or partition and the histograms of a particular interesting column are important in many ways. One of the key use cases of statistics is 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. Statistics may sometimes meet the purpose of the users' queries. Users can quickly get the answers for some of their queries by only querying stored statistics rather than firing long-running execution plans.

Recent versions of Hive store table statistics in the Hive metastore. You get this for free when a table is loaded, unless you turn the table statistics off.

For example the following Hive command allows you to request the statistics for a specific table:

```
hive> describe formatted megacorp5m;
OK
# col_name              data_type               comment
mdate                   double
datebyyear              double
datebymonth             double
…
# Detailed Table Information
Database:               default
Owner:                  hpauser1
CreateTime:             Wed Jan 13 04:44:37 EST 2016
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://sashdp01.race.sas.com:8020/apps/hive/warehouse/megacorp5m
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   true
        SAS OS Name             Linux
        SAS Version             9.04.01M3P06242015
        numFiles                8
        numRows                 0
        rawDataSize             0
        totalSize               1941852320
        transient_lastDdlTime   1452678278

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
…
Time taken: 0.474 seconds, Fetched: 82 row(s)
```

However, the stats might not be correct. As we can see, the output of the Hive command returns 0 rows (although the table has around 5 million rows). In this case, if you want the real statistics then you first need to run another command to build them:

```
hive> analyze table megacorp5m compute statistics;
Query ID = hpauser1_20160413080404_65da088c-0a2c-4bd9-93b9-32163c94dfb8
Total jobs = 1
Launching Job 1 out of 1
…
Table default.megacorp5m stats: [numFiles=8, numRows=4522868, totalSize=1941852320,
rawDataSize=1937329452]
OK
Time taken: 26.144 seconds
```

Now, if you run the described formatted command another time, you will have the real row count:

```
…
Table Parameters:
        COLUMN_STATS_ACCURATE   true
        SAS OS Name             Linux
        SAS Version             9.04.01M3P06242015
        numFiles                8
        numRows                 4522868
        rawDataSize             1937329452
        totalSize               1941852320
        transient_lastDdlTime   1460549071
….
```

So you might consider running ANALYZE commands in batch mode to keep your stats updated, so that you can benefit from this pre-computed information in your queries. Alternatively you might consider using specific formats such as ORC, Parquet, Avro, and other formats, which are natively storing these kind of aggregates.

1.7.2.2 READ_METHOD choice

So if the statistics are available in Hive one of the benefits to using the statistics would be to determine the best way for SAS to get to the data in Hive.

There are two options: JDBC or HDFS.

JDBC specifies that data is to be read through the JDBC connection to the Hive service. **HDFS** specifies that data is to be read through a connection to the Hadoop HDFS service (most of the time using a temporary table via the CREATE TABLE AS SELECT order).

READ_METHOD=JDBC will reduce performance if you are reading a large amount of data from Hive into SAS. But for small tables (<10,000 rows) it can be a better choice. It has the advantage of avoiding the CTAS and doing a direct fetch on the small result set.

In conclusion if there are a lot of CTAS operation on small tables, then using READ_METHOD=JDBC can improve the overall performance.

2 In-Hadoop processing

2.1 Introduction

Using SAS in-database processing, you can run scoring models, some SAS procedures, DS2 thread programs, and formatted SQL queries directly inside the data store.

Most of in-database processing is leveraged by the SAS Embedded Process.

What is the SAS Embedded Process for Hadoop doing? And in which cases will the embedded process will be used to generate jobs in Hadoop?

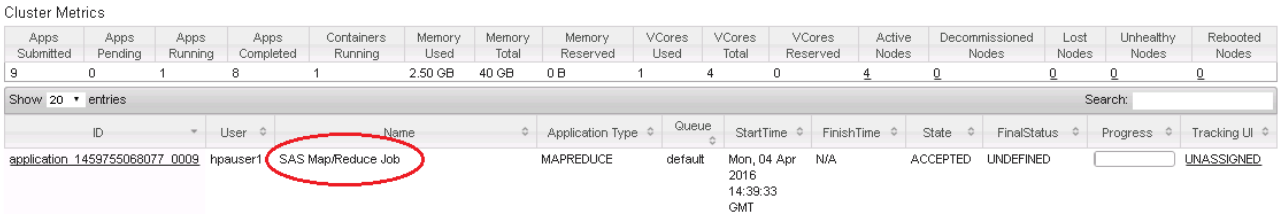
The SAS Embedded Process for Hadoop will convert a SAS operation (or a part of it) running on Hadoop data in either a MapReduce or Spark job. These SAS operations are:

- DS2 code with the DS2ACCEL option (custom DS2 code or code that is generated by SAS Data Quality Accelerator, SAS Scoring Accelerator, and so on)
- HPA procedures (the data lifting part)
- Parallel loading into SAS LASR Analytic Server (HPDS2)
- WHERE requests on SPDE files stored in Hadoop

Most of the SAS Embedded Process processing is triggering **MapReduce** jobs, but with the latest release of SAS Data Loader for Hadoop, some specific data integration and data quality operations can be executed with the Spark framework instead of MapReduce (See SAS Data Management Accelerator for Spark.)

How do I know that the SAS Embedded Process is used?

Most of the time we can see it from the Resource Manager console, when the SAS MapReduce job appears as the name of the Hadoop job.



Or when using the SAS Data Loader for Hadoop data quality features, the Resource Manager will display something like in the screenshot below (row 1 and 3):

application_1460493005517_0035	gertob	sas.dm.sparkrunner.SparkRunner	SPARK	root.gertob	Thu Apr 21 16:31:10 +0200 2016	Thu Apr 21 16:31:28 +0200 2016	FINISHED
application_1460493005517_0034	gertob	oozie:launcher:T=shell:W=SAS Spark Runner:A=spark-action-1:ID=0000009-160412163330717-oozie-oozi-W	MAPREDUCE	root.gertob	Thu Apr 21 16:30:50 +0200 2016	Thu Apr 21 16:31:30 +0200 2016	FINISHED
application_1460493005517_0033	gertob	sas.dm.sparkrunner.SparkRunner	SPARK	root.gertob	Thu Apr 21 16:29:59 +0200 2016	Thu Apr 21 16:30:20 +0200 2016	FINISHED

A recent paper from SAS R&D that was presented at the SAS Global Forum 2016 (Exploring SAS® Embedded Process Technologies on Hadoop®) provides all the details on how SAS technologies are running inside the Hadoop framework with SAS In-Database Coding Accelerator and SAS Scoring Accelerator for Hadoop.

2.2 Push processing down to Hadoop!

2.2.1 Running simple procedures in-Hadoop

As of date, the SAS procedures that can be automatically converted in advanced SQL to run in-Hadoop are:

- PROC FREQ (Use SAS/ACCESS to generate the in-database SQL code. Hive will generate the MapReduce job.)
- REPORT (Use SAS/ACCESS to generate the in-database SQL code. Hive will generate the MapReduce job.)
- SUMMARY/MEANS (Use SAS/ACCESS to generate the in-database SQL code. Hive will generate the MapReduce job.)
- TABULATE (Use SAS/ACCESS to generate the in-database SQL code. Hive will generate the MapReduce job.)
- TRANSPOSE (Use the SAS Embedded Process to generate a SAS MapReduce job.)

The SQLGENERATION option specifies whether and when SAS procedures generate SQL for in-database processing of source data.

(See: http://support.sas.com/documentation/cdl/en/acreldb/68028/HTML/default/viewer.htm#p0ltbj6gvz6w1sn1oqg2cq15ff6u.htm)

By default this option is set to: (NONE DBMS='TERADATA DB2 ORACLE NETEZZA ASTER GREENPLM HADOOP SAPHANA IMPALA HAWQ')

As HADOOP is in the list, it means that SQL code for in-database (or in-Hadoop) processing will be generated when you use those procedures with SAS/ACCESS to Hadoop LIBNAME statements.

With the appropriate SAS tracing options, we can see in the log this kind of message for a PROC FREQ:

```
SQL_IP_TRACE: Some of the SQL was directly passed to the DBMS.
NOTE: SQL generation will be used to construct frequency and crosstabulation
tables.
```

It means that the SAS/ACCESS engine will be able to build the Hadoop-compliant SQL code to create cross-tabulation tables required by PROC FREQ. The SQL code will be sent to Hive (or Tez or Impala), which will generate the MapReduce job that will run in parallel on all the Hadoop nodes.

If the SQLGENERATION is NOT set, then a temporary table with all required columns will be created in Hive, then downloaded to the SAS server for local execution of the procedure.

The bigger your table is, the more efficient the in-database processing will be.

Note: This feature is not available for HDMD or SPDE in HDFS tables.

2.2.2 Running DS2 code in-Hadoop

DS2 is a new SAS proprietary programming language that is appropriate for advanced data manipulation. DS2 is included with Base SAS and intersects with the SAS DATA step.

DS2 takes advantage of threaded processing, so it can bring significant performance gains running directly inside distributed databases via the SAS In-Database Code Accelerator.

You can submit multi-threaded DS2 code through the Base SAS language interface locally on the Base SAS machine (multi-core processing on one machine), but you can also run a DS2 program directly to a data source using SAS In-Database Code Accelerator (multi-core on multiple nodes processing).

The DS2ACCEL= system option controls whether DS2 code is executed inside the database. (See http://support.sas.com/documentation/cdl/en/ds2ref/68052/HTML/default/viewer.htm#p0f0lxzn6nt6xin10yvi3e384pnu.htm)

The default value is NONE, which disables the DS2 code from executing in a supported parallel environment.

Make sure that DS2ACCEL=ANY is specified if you want to leverage in-database processing when you submit DS2 code.

Both the DS2 data and thread program can run inside the database if the output table from the data program resides in Hadoop.

(See: http://support.sas.com/documentation/cdl/en/indbug/68442/HTML/default/viewer.htm#p09t4neh55n034n1ssg584lhlco7.htm for details.)

Example (with 5-million-row table):

DS2 data and thread programs

```
%let wtab=megacorp5m;
proc ds2 bypartition=yes;
thread t_pgm / overwrite=yes;
   drop regionlongitude regionlatitude statelongitude statelatitude
citylongitude citylatitude
   ;
   dcl char(20) productdesc_clean;
   dcl double y;
   method run();
      set hivelib.&wtab;
      productdesc_clean = strip(productdescription);
        /*date is a reserved key word  -rename date in mdate to make the
formula work*/
        y = year(datepart(mdate));
```

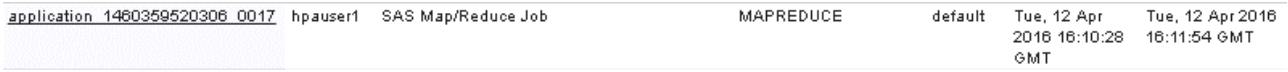
```
   end;
endthread;

data hivelib.&wtab.Bis(overwrite=yes);
   declare thread t_pgm t;
   method run();
      set from t;
   end;
enddata;
run;
quit;
```

No DS2ACCEL option set	With DS2ACCEL=ANY
No MapReduce Job, the DS2 proc runs in multi-threaded mode on the SAS Server.	SAS Map Reduce Job is generated and launched.
NOTE: PROCEDURE DS2 used (Total process time): real time 7:35.94 cpu time 3:53.35	NOTE: Created thread t_pgm in data set work.t_pgm. NOTE: Running THREAD program in-database NOTE: Running DATA program in-database NOTE: Execution succeeded. No rows affected. 48 quit; NOTE: PROCEDURE DS2 used (Total process time): real time 1:40.40 cpu time 0.44 seconds

Note the massive difference in CPU time.

When the DS2ACCEL=ANY is set, a SAS MapReduce job is directly generated by the SAS Embedded Process:



2.2.3 Running DATA steps in-Hadoop

If the SAS Embedded Process is installed in the Hadoop cluster, it is possible for DATA step code to be executed in parallel against the input data residing on the HDFS file system.

To enable the DATA step to be run inside Hadoop, set the DSACCEL= system option to ANY.

The DATA step has been enhanced to determine when the user code is appropriate for exporting to the Hadoop MapReduce facility.

The DATA step can be run inside Hadoop under the following conditions:

- Only one input file and one output file are allowed.
- The input file **and** the output file are in Hadoop.
- Only functions and formats that are supported by the DS2 language compile successfully.
- Some DATA step statements are not allowed, such as those pertaining to input and output.

As of date, there still many restrictions on which DATA step features are eligible for the MapReduce conversion. Review Requirements for DATA Step Processing and Restrictions in DATA Step Processing.

If a SAS program does not meet the requirements for running in Hadoop, the code executes in your Base SAS session. In this case, SAS reads and writes large tables over the network.

A best practice is to use a sample table and MSGLEVEL to determine if your DATA step is compliant for in-Hadoop processing before running the DATA step code on the real potentially big table.

Example of unsuccessful attempt:

```
26          if _N_=1;
                    _
                    827
INFO: DATA Step contains subsetting-if.

27          if UnitReliability >0.95 then score=0; Else score=1;
28          run;

INFO: Could not run DATA Step in HADOOP.
```

Example of successful execution with the SAS In-Database Code Accelerator:

Code:

```
data hivelib.megacorp_score;
set hivelib.megacorp5m;
if UnitReliability >0.99 then score=0; Else score=1;
run;
```

Log:

```
NOTE: Attempting to run DATA Step in Hadoop.
NOTE: DATA Step code for the data set "HIVELIB.MEGACORP_SCORE" was executed in
the Hadoop EP environment.

(HDP_JOB_ID), job_1460359520306_0018, SAS Map/Reduce Job,
http://sashdp01.race.sas.com:8088/proxy/application_1460359520306_0018/
```

```
         Hadoop Job (HDP_JOB_ID), job_1460359520306_0018, SAS Map/Reduce Job,

http://sashdp01.race.sas.com:8088/proxy/application_1460359520306_0018/
         Hadoop Version       User
CREATE TABLE `MEGACORP_SCORE` (`MDATE` DOUBLE,`DATEBYYEAR` DOUBLE,`DATEBYMONTH`
DOUBLE,`DAYOFWEEK` DOUBLE,`FACILITYID`

DOUBLE,`FACILITY` VARCHAR(7),`FACILITYTYPE` VARCHAR(5),`FACILITYAGE`
…
DOUBLE,`B_BRAND` DOUBLE,`B_FAILURE` DOUBLE,`SCORE` DOUBLE) ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001' LINES TERMINATED BY
'\012' STORED AS TEXTFILE TBLPROPERTIES ('SAS OS Name'='Linux','SAS
Version'='9.04.01M3P06242015')

2.6.0.2.2.0.0-2041           hpauser1


         Started At                 Finished At
HADOOP_191: Executed: on connection 4
         Apr 12, 2016 3:08:19 PM    Apr 12, 2016 3:09:35 PM
LOAD DATA INPATH '/tmp/megacorp_score_2016-04-12-15-08-18-997-02' OVERWRITE INTO
TABLE `MEGACORP_SCORE`
```

Reference:
http://support.sas.com/documentation/cdl/en/indbug/68442/HTML/default/viewer.htm#p0nvqb8emmd9vvn170j4ijyczmgx.htm

2.3 File formats & partitioning

Alternative file formats in Hive and partitioning options are covered in the previous section. However it is important to note that these capabilities are not only available in the SAS/ACCESS engine but also for the SAS In-Database Code Accelerator.

In the February 2015 release of SAS 9.4, the following changes and enhancements were made. The SAS In-Database Code Accelerator for Hadoop uses HCatalog to process complex, non-delimited files. This enables the SAS In-Database Code Accelerator for Hadoop to support Avro, ORC, RCFile, and Parquet file types.

Limitations:

As of date, HCatalog file formats are not supported on Pivotal HD v2.x or IBM BigInsights v3.x and later, and partitioned Parquet data is currently not supported as input to the SAS In-Database Code Accelerator for Hadoop.

See:

http://support.sas.com/documentation/cdl/en/indbug/68442/HTML/default/viewer.htm#n0sutvfq0qpa1jn12xci02u1yypq.htm

2.4 Resource management & Hadoop tuning

When SAS processing is run inside Hadoop via the SAS Embedded Process, it integrates as a MapReduce application that will be scheduled and controlled by YARN.

While you will benefit from fully distributed SAS processing across your Hadoop cluster, it is very important to understand how MapReduce settings can impact SAS Embedded Process jobs running in YARN.

For example, if your tuning is inappropriate, and the MapReduce job generated by the SAS Embedded Process is using more resources than originally scheduled, then YARN can kill this process and your SAS job will fail.

mapreduce.map.memory.mb	Memory size for the map task container.
mapredure.map.java.opts	Heap size for child JVMs of maps.
mapreduce.reduce.memory.mb	Memory size for the reduce task container.
mapreduce.reduce.java.opts	Heap size for child JVMs of reduce.
mapreduce.job.reduce.slowstart.completedmaps	Fraction of the number of maps in the job that should be complete before reduces are scheduled for the job. (Default behavior is to launch reduces well before all mappers complete.)
mapreduce.job.queuename	Specifies the YARN queue for all MapReduce applications used by this client.

The NodeManager can monitor the memory usage (virtual and physical) of the container.

When the YARN property ***yarn.nodemanager.pmem-check-enabled*** is true (by default in recent Hadoop versions) it causes YARN to kill a container that is using more physical memory than the expected memory size ("mapreduce.reduce.memory.mb" or "mapreduce.map.memory.mb").

If the process is killed, then SAS code can only provide a generic failed message because it can't determine if the failure was because of a resource issue (YARN killing it) or if the process actually had a problem and crashed.

This paper does not cover the resource management topic for SAS processing in Hadoop, as best practices have already been provided by SAS through several papers:

- Best Practices for Resource Management in Hadoop
- SAS® Analytics on Your Hadoop Cluster Managed by YARN

Those papers provide YARN and MapReduce tuning advice for SAS Embedded Process jobs (but also for SAS/ACCESS and in-memory processing, even when all of them coexist in a shared Hadoop cluster). The papers explore real use cases.

3 Lessons learned

3.1 SAS on Hadoop general lessons learned

Most of this list comes from proof of concept projects or project experiences shared by SAS offices in Belgium, Portugal, and Philippines.

Most of the topics listed below are discussed in this document.

- **Fine-tune the Hadoop cluster**
  - I/O processing
  - Network speed
  - Engage the Hadoop vendor services to get support on MapReduce and YARN tuning.
- **Use formats in Hadoop tables** (see Changing the default maximum length for SAS character columns)
- **Use SAS Data Integration Studio for better overview and control**
- **Manage job priorities in YARN** (see YARN commands)
- **Use SAS Data Loader for prototyping and working on sample data**
- **Use partitioning in Hadoop with low cardinality and often queried variables** (see Partitioning)
- **Be aware that there is a minimum required size of data before Hadoop will make a difference**
- **Be aware of what Hive is not (Hive limits)** (see Optimize your SELECT and The "Order by" hurdle)
- **Use the SAS Data Integration Studio Hive transformation as much as possible (making sure the code is in-database)**
- **Focus on the code of heavy data processing, making sure it is Hadoop in-database** (see Push data management down to Hadoop! and Push processing down to Hadoop!)
- **Use Hue for debugging purposes** (SAS trace does not give you the full debugging information.)

3.2 SAS jobs adaptation lessons learned

The following recommendations are useful for traditional SAS data integration or for adapting and optimizing SAS jobs to work with data inside Hadoop.

- **Eliminate code that is useless on the Hadoop platform,** such as PROC SORT before PROC MEANS, and remove SAS indexes.
- **The DELETE statement needs to be used before the creation of tables** as Hadoop doesn't support the REPLACE TABLE statement.

- **Look for SAS functions that are not available on Hadoop and change them,** for example, sum(0,var), rank, intnx, and put (see Best practices).
- **Use DDL ALTER TABLE statement to adapt the columns.** SAS/ACCESS reads Hive data type strings with a default length of 32K (see Changing the default maximum length for SAS character columns).
- **If using the Hive 0.13 and Impala, the UPDATE statement is not supported.** (It is available only on Hive 0.14.)
- **Consider using the Hadoop CAST function to readapt date columns** (hive-date, impala-timestamp). SAS might not properly convert numeric values in the SAS data format to all date types in Hadoop.

These lessons are provided from the SAS office in Portugal, based on a recent SAS on Hadoop project. They also provided the to-do list presented in the next section.

3.3 Make your SAS jobs ready for Hadoop

The to-do list below describes actions required to make traditional SAS jobs (working with database management systems or SAS data sets) more Hadoop friendly.

- Eliminate SAS indexes.
- Remove unnecessary PROC SORT statements before each PROC MEANS statement when working with Hadoop data instead of SAS data sets.
- Drop the Table Loader transformations that only recreate the indexes.
- Adapt PROC SORT code to SQL syntax.
- Use the DELETE statement to replace tables.
- Use Hive transformations and explicit code.
- Adapt statements that use Boolean variables.
- Adapt functions such as rank, put(var, format), sum(0,var), intnx.
- Adapt SAS macro code.
- Look for and review DO WHILE statements that yield multiple outputs.
- Recode SAS specific statements such as set/key (iorc), PROC SORT NODUPKEY

3.4 Project use case

3.4.1 Typical performances issues (project example)

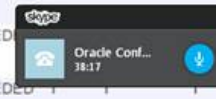
When working with SAS Data Integration Studio on a large Hadoop implementation for a Telco customer project, the following behaviors were identified by the Hadoop vendor (Cloudera) consultant as impacting the performance. See his notes below:

1) Problem: Performance of data loading.

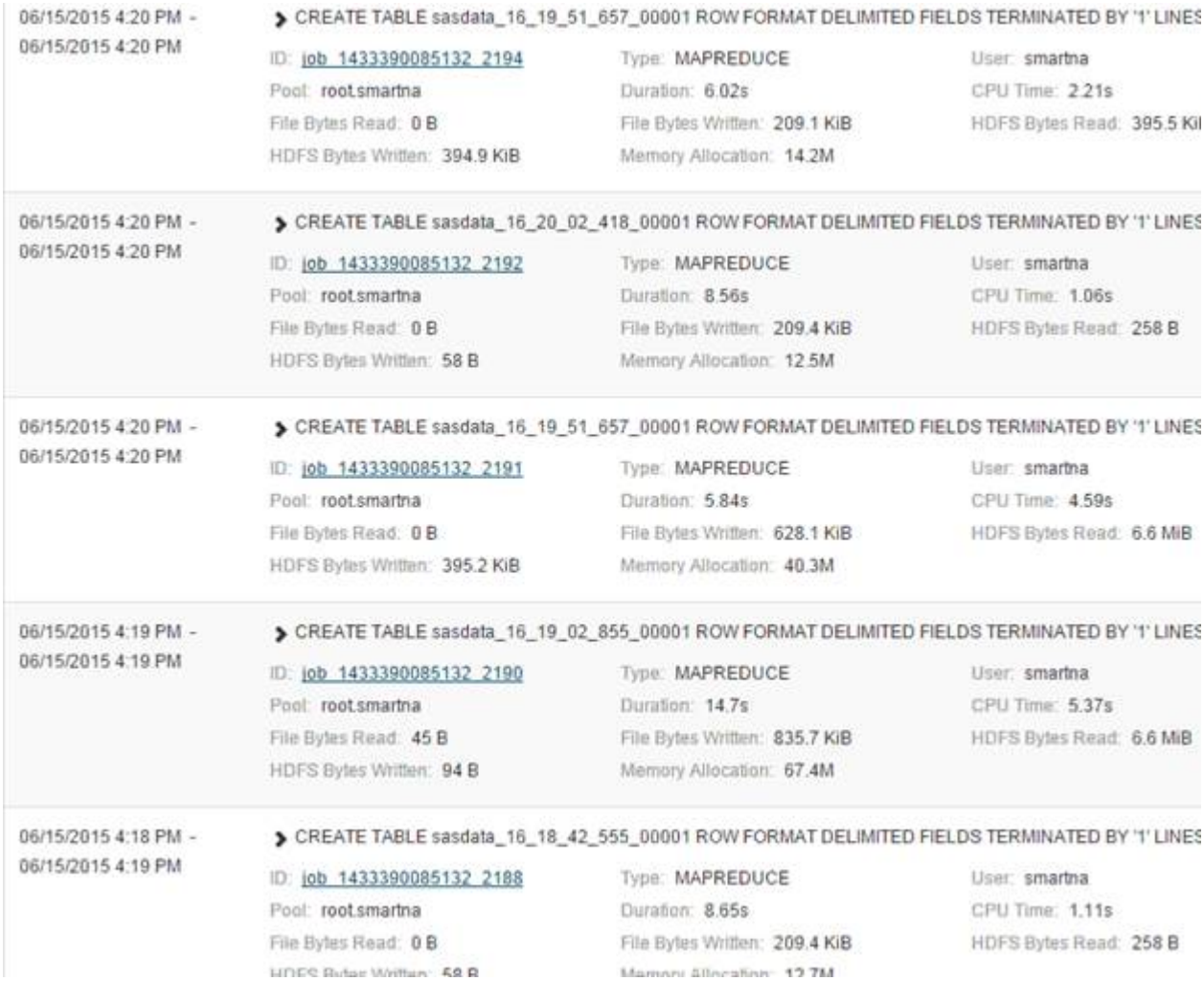
Summary: *Customer loads data from an external database to HDFS through SAS Data Integration Studio. This tool generates some workload. A lot of CREATE TABLE AS SELECT statements are generated when the data load jobs are simply reading an Oracle table and loading its contents into an existing Cloudera Hive partitioned table.*

Root of the problem: *A lot of small MapReduce jobs that run concurrently.*

2015.06.15 16:37:13 PHT	2015.06.15 16:37:23 PHT	2015.06.15 16:37:37 PHT	job_1433390085132_2208	CREATE TABLE sasdata_16_37_12_946_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1
2015.06.15 16:36:37 PHT	2015.06.15 16:36:46 PHT	2015.06.15 16:37:01 PHT	job_1433390085132_2207	CREATE TABLE sasdata_16_36_36_854_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1
2015.06.15 16:36:17 PHT	2015.06.15 16:36:27 PHT	2015.06.15 16:36:42 PHT	job_1433390085132_2206	CREATE TABLE sasdata_16_36_17_059_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1
2015.06.15 16:36:10 PHT	2015.06.15 16:36:19 PHT	2015.06.15 16:36:33 PHT	job_1433390085132_2205	CREATE TABLE sasdata_16_36_09_411_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1
2015.06.15 16:35:47 PHT	2015.06.15 16:35:57 PHT	2015.06.15 16:36:13 PHT	job_1433390085132_2204	CREATE TABLE sasdata_16_35_47_546_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1
2015.06.15 16:34:29 PHT	2015.06.15 16:34:39 PHT	2015.06.15 16:34:56 PHT	job_1433390085132_2203	CREATE TABLE sasdata_16_34_28_942_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1
2015.06.15 16:31:25 PHT	2015.06.15 16:31:35 PHT	2015.06.15 16:31:51 PHT	job_1433390085132_2202	CREATE TABLE sasdata_16_31_25_425_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1
2015.06.15 16:30:39 PHT	2015.06.15 16:30:49 PHT	2015.06.15 16:31:07 PHT	job_1433390085132_2201	CREATE TABLE sasdata_16_30_39_488_00...TXT_1(Stage	smarthpa	root.smarthpa	SUCCEED			
2015.06.15 16:30:38 PHT	2015.06.15 16:30:48 PHT	2015.06.15 16:31:04 PHT	job_1433390085132_2200	CREATE TABLE sasdata_16_30_37_482_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED			
2015.06.15 16:30:30	2015.06.15 16:30:39	2015.06.15 16:30:55	job_1433390085132_2199	CREATE TABLE sasdata_16_30_29_744_00...TXT_1(Stage	smartna	root.smartna	SUCCEEDED	1	1	1



Each job has one mapper and produces a very small amount of data:



Also, each job creates a new table. That means separate DML operations in the Hive metastore (inserts or updates in the database).

Possible solutions:

- Rewrite (or reconfigure) the application to use fewer big jobs instead of many small jobs.
- Use INSERT SELECT statements instead CREATE SELECT statements.

2) Problem: Some queries go slowly.

Root of the problem: *They use only one reducer.*

3.4.2 Analysis and recommendations

One way to address the first problem is to delegate the external DBMS loading in Hadoop to bulk load tools such as Sqoop. (Note that SAS Data Loader for Hadoop provides a directive to use Sqoop.)

Another recommendation is to check whether statistics collection is enabled in SAS Data Integration Studio. If so, try turning it off, which will eliminate automatic COUNT (*) operations. Finally, to optimize access to many small tables, consider using the READ_METHOD=JDBC (see READ_METHOD choice).

For the second problem, it is possible to influence the number of reducers that will be used in the MapReduce job via the PROPERTIES LIBNAME option, for example:

```
properties= mapreduce.job.reduces=12
```

However the unique reducer is more likely coming from an ORDER BY in the SAS generated SQL. (See The “Order by” hurdle).

4 Monitoring SAS in Hadoop

4.1 Increase SAS verbosity

Warning: Consultants should switch off verbose logging once they have completed their troubleshooting as all that logging can cause large unwanted files.

4.1.1 SASTRACE, SQL_IP_TRACE and MSGLEVEL

- The **SASTRACE** option can be used in SAS code and allows you to know where the processing will take place.

Example:

```
OPTION SASTRACE=',,,ds' SASTRACELOC=SASLOG NOSTSUFFIX;
<insert the code you want to trace here>
OPTIONS SASTRACE=off;
```

The SAS log will contain each SQL command issued and executed, prefaced with a database-specific entry like HADOOP_n, where n is an integer indicating the number-in-sequence of the SQL statement as it occurred in that SAS session.
The s allows you to have a summary statistics table at the end of log:

```
Summary Statistics for HADOOP are:
Total SQL execution seconds were:              0.465640
Total SQL prepare seconds were:               31.498703
Total SQL describe seconds were:               0.063854
Total seconds used by the HADOOP ACCESS engine were    32.949886
```

- The **MSGLEVEL=I** option prints additional notes in the SAS log pertaining to index usage, merge processing, and sort utilities, along with standard notes, warnings, CEDA message, and error messages.

```
OPTION MSGLEVEL=I;
```

You can determine whether your code is non-compliant for Hadoop by setting the system option MSGLEVEL=I. When MSGLEVEL=I, SAS writes log messages that identify the non-compliant code.

Since the July 2015 release for SAS 9.4, when the MSGLEVEL=I option is set and a job fails, a link to the HTTP location of the MapReduce logs is also produced.
Here is an example:

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

- The **SQL_IP_TRACE** option specifies that a note will be written to the log each time the SQL is modified by SAS and submitted to the DBMS. Each time you receive the message that indicates that the access engine is changing the original SQL code so that it can be processed by the DBMS.

For procedures FREQ, MEANS, SUMMARY, TABULATE, and REPORT (which all generate generic SQL code), the undocumented SQL_IP_TRACE system option can reveal the SQL query that they generate. This option is intended to reveal operational details of the SQL implicit pass-through facility. When set to SOURCE, the SQL_IP_TRACE option causes the generic query generated by an in-database-enabled procedure to be printed to the SAS log.

Note: You can combine all these options in a single line:

```
/*Trace options*/
OPTION SASTRACE=',,,ds' SASTRACELOC=SASLOG NOSTSUFFIX
SQL_IP_TRACE=(note, source) msglevel=i;
```

4.1.2 EP Trace

Follow this procedure to enable additional tracing for the SAS Embedded Process.

- Add the following block in the mapred-site.xml on the client side.

For example in: /opt/sas/hadoop/conf/mapred-site.xml:

```
<property>
   <name>sas.ep.server.trace.level</name>
   <value>10</value>
</property>
```

- After running the job, from the account used to run the application, log in to one of the nodes of the cluster so that you can retrieve all the container logs by issuing the following:

```
hpauser1@sashdp01~]$ yarn logs -applicationId <YARN_APPLICATION_ID>
```

Here's an example:

```
hpauser1@sashdp01~]$ yarn logs -applicationId application_1456785111075_0001
```

Example:

```
(CFG)  sas.ep.input.metadata=/tmp/megacorp5m_4a38e051-51f4-5848-b465-d348a1bbe0fd/megacorp5m.xml
20160502:12.22.24.67: 00000004:            DS2 : [
table gridtf.out;dcl double "UnitDowntime"; keep "UnitDowntime";dcl double "UnitAge"; keep "UnitAge";dcl double "UnitLifespanL
imit"; keep "UnitLifespanLimit";dcl double "UnitReliability"; keep "UnitReliability";dcl double "UnitCapacity"; keep "UnitCapa
city";dcl char(7) CHARACTER SET "latin1" "Facility"; keep "Facility";dcl char(5) CHARACTER SET "latin1" "FacilityType"; keep "
FacilityType";dcl double "UnitLifespan"; keep "UnitLifespan";method run();set sasep.in;output;end;endtable;
]
20160502:12.22.24.71: 00000004:NOTE: Task type            [1]
20160502:12.22.24.71: 00000004:NOTE: Trace Level          [10]
20160502:12.22.24.71: 00000004:NOTE: Map Task Count       [16]
20160502:12.22.24.71: 00000004:NOTE: Session CEI          [0]
20160502:12.22.24.71: 00000004:NOTE: Output Delimiter     [0]
20160502:12.22.24.71: 00000004:NOTE: Output Text Qualifier [0]
20160502:12.22.24.71: 00000004:NOTE: Output Record Format  [1]
20160502:12.22.24.71: 00000004:NOTE: GridParms: Mode      [input ]
20160502:12.22.24.71: 00000004:NOTE: GridParms: Port      [61000]
20160502:12.22.24.71: 00000004:NOTE: GridParms: Nodes     [4]
20160502:12.22.24.71: 00000004:NOTE: GridParms: Blk Factor [10000]
20160502:12.22.24.71: 00000004:NOTE: GridParms: Dist Port  [35484]
20160502:12.22.24.71: 00000004:NOTE: GridParms: Num Vars   [8]
20160502:12.22.24.71: 00000004:NOTE:              Name=[UnitDowntime] CharLen=[0] ByteLen=[-8] CEI=[0]
20160502:12.22.24.71: 00000004:NOTE:              Name=[UnitAge] CharLen=[0] ByteLen=[-8] CEI=[0]
20160502:12.22.24.71: 00000004:NOTE:              Name=[UnitLifespanLimit] CharLen=[0] ByteLen=[-8] CEI=[0]
20160502:12.22.24.71: 00000004:NOTE:              Name=[UnitReliability] CharLen=[0] ByteLen=[-8] CEI=[0]
20160502:12.22.24.71: 00000004:NOTE:              Name=[UnitCapacity] CharLen=[0] ByteLen=[-8] CEI=[0]
20160502:12.22.24.71: 00000004:NOTE:              Name=[Facility] CharLen=[7] ByteLen=[7] CEI=[29]
20160502:12.22.24.71: 00000004:NOTE:              Name=[FacilityType] CharLen=[5] ByteLen=[5] CEI=[29]
20160502:12.22.24.71: 00000004:NOTE:              Name=[UnitLifespan] CharLen=[0] ByteLen=[-8] CEI=[0]
20160502:12.22.24.71: 00000004:NOTE: Output File Type      [DELIMITED]
20160502:12.22.24.71: 00000004:NOTE: Output Directory      [hdfs://sashdp01.race.sas.com:8020/tmp/megacorp5m_4a38e051-51f4-584
8-b465-d348a1bbe0fd/megacorp5m]
20160502:12.22.24.71: 00000004:NOTE: Job ID                [job_1462175209030_0002_attempt_1462175209030_0002_m_000014_0]
20160502:12.22.24.71: 00000004:           Allocated INPUT BufferInfo array with space for [4] items and [1] writers.
20160502:12.22.24.71: 00000004:           Allocated OUTPUT BufferInfo array with space for [3] items and [1] writers.
20160502:12.22.24.74: 00000005:         > runAction
20160502:12.22.24.75: 00000006:         > processEmbeddedProcessAction Action=[1]
20160502:12.22.24.75: 00000006:         > getTKTSConnection
20160502:12.22.24.75: 00000006:           Grid hosts=[1038436736] Grid Host Offset=[5]
20160502:12.22.24.75: 00000006:           Number of HPA captains=[4]
20160502:12.22.24.75: 00000006:           Number of HPA connections per captain=[5]
20160502:12.22.24.75: 00000006:           Number of Map tasks=[16]
20160502:12.22.24.75: 00000006:           Connection String=[DRIVER=DS2;CONOPTS=(DRIVER=TSSQL;CONOPTS=((DRIVER=HadoopMR;CATALO
G=sasep);(DRIVER=GridTS;CATALOG=gridtf;PORT=140562433007408))))]
20160502:12.22.24.81: 00000006:WARNING: [01S02]Current catalog set to SASEP (0x80fff8bd)
20160502:12.22.24.81: 00000006:         < getTKTSConnection rc=[0]
20160502:12.22.24.81: 00000006:         > prepareDS2Code
20160502:12.22.24.87: 00000006:         > HadoopMRPrepare
20160502:12.22.24.87: 00000006:           Statement type is INPUT SELECT.
20160502:12.22.24.87: 00000006:         > Hadoop_InSelectStmt
20160502:12.22.24.87: 00000006:           INPUT Columns definition:
20160502:12.22.24.87: 00000006:             Ordinal/Name   = [1][mdate][5]
```

Note: Be careful when you turn on the SAS Embedded Process as it generates a very verbose log. For example, a SAS MapReduce job for a simple high-performance analytical procedure can generate a 37MB file (instead of a 250KB file, which is generated with standard logging).

4.1.3 GridDriver traces

If you are using the TKGrid in coordination with the SAS Embedded process (for example, for parallel loading to SAS LASR Analytic Server or for executing high-performance analytics procedures), for troubleshooting purposes you might need to activate the associated trace.
Follow this procedure to enable this additional tracing for the SAS Embedded process.

- Make a local copy of the SAS Embedded Process configuration file (stored in HDFS)

```
[hdfs@sashdp01 ~]$ hadoop fs -get /sas/ep/config/ep-config.xml
[hdfs@sashdp01 ~]$ ll
total 4
-rw-r--r-- 1 hdfs hadoop 2571 Apr 15 01:16 ep-config.xml
[hdfs@sashdp01 ~]$ cp ep-config.xml ep-config.xml.orig
```

```
[hdfs@sashdp01 ~]$ vi ep-config.xml
```

- Set the EP trace level property to 11. This will force the "grid driver" to dump its trace information. Add the block at the bottom of the XML file (using VI editor for example):

```
<property>
    <name>sas.ep.server.trace.level</name>
    <value>11</value>
  </property>
```

- Then upload the file in HDFS.

```
[hdfs@sashdp01 ~]$ hadoop fs -rm /sas/ep/config/ep-config.xml
16/04/15 01:18:57 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion
interval = 360 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://sashdp01.race.sas.com:8020/sas/ep/config/ep-config.xml' to trash at:
hdfs://sashdp01.race.sas.com:8020/user/hdfs/.Trash/Current
[hdfs@sashdp01 ~]$ hadoop fs -put ep-config.xml /sas/ep/config/
```

- Check or create a /opt/SAS folder on all nodes.

The "grid driver" writes its traces to /opt/SAS. That folder needs to exist on all nodes where the SAS Embedded Process is installed. Its permissions need to be set to 777.

So you'll need to create /opt/SAS on all nodes (make sure the folder /opt/SAS does not already exist!!!) and set the directory permissions to 777:

```
[root@sashdp01 ~]# for hst in `cat /etc/gridhosts`; do ssh -q $hst "hostname;mkdir
/opt/SAS;chmod 777 /opt/SAS";done
```

- After the reproducing the problem, check if there are log messages:

```
[root@sashdp01 ~]# for hst in `cat /etc/gridhosts`; do ssh -q $hst "hostname;ls -alrt
/opt/SAS";done
sashdp01.race.sas.com
total 8
drwxr-xr-x. 10 root root 4096 Apr 12 09:54 ..
drwxrwxrwx   2 root root 4096 Apr 12 09:54 .
sashdp02.race.sas.com
total 8
drwxr-xr-x. 11 root root 4096 Apr 12 09:54 ..
drwxrwxrwx   2 root root 4096 Apr 12 09:54 .
sashdp03.race.sas.com
total 8
drwxr-xr-x. 10 root root 4096 Apr 12 09:54 ..
drwxrwxrwx   2 root root 4096 Apr 12 09:54 .
sashdp04.race.sas.com
total 3484
drwxr-xr-x. 8 root root      4096 Apr 12 09:54 ..
drwxrwxrwx. 2 root root      4096 Apr 15 12:05 .
-rw-r--r--. 1 yarn hadoop 3556462 Apr 15 12:06 SASmsg
```

Reference:
http://esurveys.na.sas.com/TESSA/main/searchResultList.jsp?qt=7611727278&qtpure=7611727278&isQuickSearch=&collection=all

4.2 Monitor SAS jobs and storage in Hadoop

There are multiple monitoring interfaces available in the Hadoop environment.

4.2.1 YARN commands

Regardless of which Hadoop vendor you use, you can use the standard Apache Hadoop utilities: the YARN command line and the Resource Manager UI (unless they were deliberately disabled by the Hadoop administrator).

- The YARN command line can be used to follow the job execution.

Tip: Type "hadoop version" to know which version of Hadoop you are running.

```
[hpauser1@sashdp01 ~]$ hadoop version
Hadoop 2.6.0.2.2.0.0-2041
Subversion git@github.com:hortonworks/hadoop.git -r
7d56f02902b436d46efba030651a2fbe7c1cf1e9
Compiled by jenkins on 2014-11-19T19:42Z
Compiled with protoc 2.5.0
From source with checksum f0c0406cc910a79f206d2ee4c2a68773
This command was run using /usr/hdp/2.2.0.0-2041/hadoop/hadoop-common-2.6.0.2.2.0.0-
2041.jar
```

Then review the Hadoop apache documentation on available YARN commands, for example, for Hadoop 2.6: https://hadoop.apache.org/docs/r2.6.0/hadoop-yarn/hadoop-yarn-site/YarnCommands.html

Among other capabilities, the YARN command allows you to monitor the progress of a job, kill it if needed, and dump the job's log.

Examples:

- Display available commands:

```
[hpauser1@sashdp01 ~]$ yarn
Usage: yarn [--config confdir] COMMAND
where COMMAND is one of:
  resourcemanager -format-state-store   deletes the RMStateStore
  resourcemanager                       run the ResourceManager
  nodemanager                           run a nodemanager on each slave
  timelineserver                        run the timeline server
  rmadmin                               admin tools
  version                               print the version
  jar <jar>                             run a jar file
  application                           prints application(s)
                                        report/kill application
  applicationattempt                    prints applicationattempt(s)
                                        report
  container                             prints container(s) report
```

```
  node                                 prints node report(s)
  cluster                              print cluster information
  queue                                prints queue information
  logs                                 dump container logs
  classpath                            prints the class path needed to
                                       get the Hadoop jar and the
                                       required libraries
  daemonlog                            get/set the log level for each
                                       daemon
 or
  CLASSNAME                            run the class named CLASSNAME
Most commands print help when invoked w/o parameters.
```

- Display the RUNNING nodes with running containers:

```
[hpauser1@sashdp01 ~]$ yarn node -list
```

```
[hpauser1@sashdp01 ~]$ yarn node -list
16/05/02 13:24:26 INFO impl.TimelineClientImpl: Timeline service address: http://sashdp01.race.sas.com:8188/ws/v1/timeline/
16/05/02 13:24:26 INFO client.RMProxy: Connecting to ResourceManager at sashdp01.race.sas.com/10.96.5.22:8050
Total Nodes:4
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
sashdp03.race.sas.com:45454            RUNNING sashdp03.race.sas.com:8042                    4
sashdp01.race.sas.com:45454            RUNNING sashdp01.race.sas.com:8042                    2
sashdp04.race.sas.com:45454            RUNNING sashdp04.race.sas.com:8042                    3
sashdp02.race.sas.com:45454            RUNNING sashdp02.race.sas.com:8042                    3
```

- Display the RUNNING applications:

```
[hpauser1@sashdp01 ~]$ yarn application -list
16/05/02 13:26:11 INFO impl.TimelineClientImpl: Timeline service address:
http://sashdp01.race.sas.com:8188/ws/v1/timeline/
16/05/02 13:26:11 INFO client.RMProxy: Connecting to ResourceManager at
sashdp01.race.sas.com/10.96.5.22:8050
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED,
RUNNING]):1
                Application-Id      Application-Name        Application-Type
User           Queue                   State           Final-State             Progress
Tracking-URL
application_1462175209030_0011  CREATE TABLE sasdata_13_24_55...`MEGACORP5M`(Stage-3)
MAPREDUCE      hpauser1      default                 RUNNING               UNDEFINED
67.43%  http://sashdp01.race.sas.com:51910
```

- Display FINISHED applications:

```
[hpauser1@sashdp01 ~]$ yarn application -list -appStates FINISHED
16/05/02 13:29:55 INFO impl.TimelineClientImpl: Timeline service address:
http://sashdp01.race.sas.com:8188/ws/v1/timeline/
16/05/02 13:29:55 INFO client.RMProxy: Connecting to ResourceManager at
sashdp01.race.sas.com/10.96.5.22:8050
Total number of applications (application-types: [] and states: [FINISHED]):13
                Application-Id      Application-Name        Application-Type
User           Queue                   State           Final-State             Progress
Tracking-URL
application_1462175209030_0007  CREATE TABLE sasdata_12_51_59...`MEGACORP5M`(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED              SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0007
```

```
application_1462175209030_0013  CREATE TABLE sasdata_13_27_26...`MEGACORP5M`(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0013
application_1462175209030_0004  CREATE TABLE sasdata_12_48_56_822_00...TXT_1(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0004
application_1462175209030_0006  CREATE TABLE sasdata_12_51_28...`MEGACORP5M`(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0006
application_1462175209030_0012  CREATE TABLE sasdata_13_26_20...`MEGACORP5M`(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0012
application_1462175209030_0005    SAS Map/Reduce Job               MAPREDUCE
hpauser1         default                FINISHED             SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0005
application_1462175209030_0011  CREATE TABLE sasdata_13_24_55...`MEGACORP5M`(Stage-3)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0011
application_1462175209030_0002    SAS Map/Reduce Job               MAPREDUCE
hpauser1         default                FINISHED             SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0002
application_1462175209030_0009  CREATE TABLE sasdata_13_24_13_700_00...TXT_1(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0009
application_1462175209030_0010  CREATE TABLE sasdata_13_24_55...`MEGACORP5M`(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0010
application_1462175209030_0008  CREATE TABLE sasdata_12_52_57...`MEGACORP5M`(Stage-1)
MAPREDUCE      hpauser1      default                 FINISHED               SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0008
application_1462175209030_0001    SAS Map/Reduce Job               MAPREDUCE
hpauser1         default                FINISHED             SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0001
application_1462175209030_0003    SAS Map/Reduce Job               MAPREDUCE
hpauser1         default                FINISHED             SUCCEEDED
100% http://sashdp02.race.sas.com:19888/jobhistory/job/job_1462175209030_0003
```

- Kill a specific job (application):

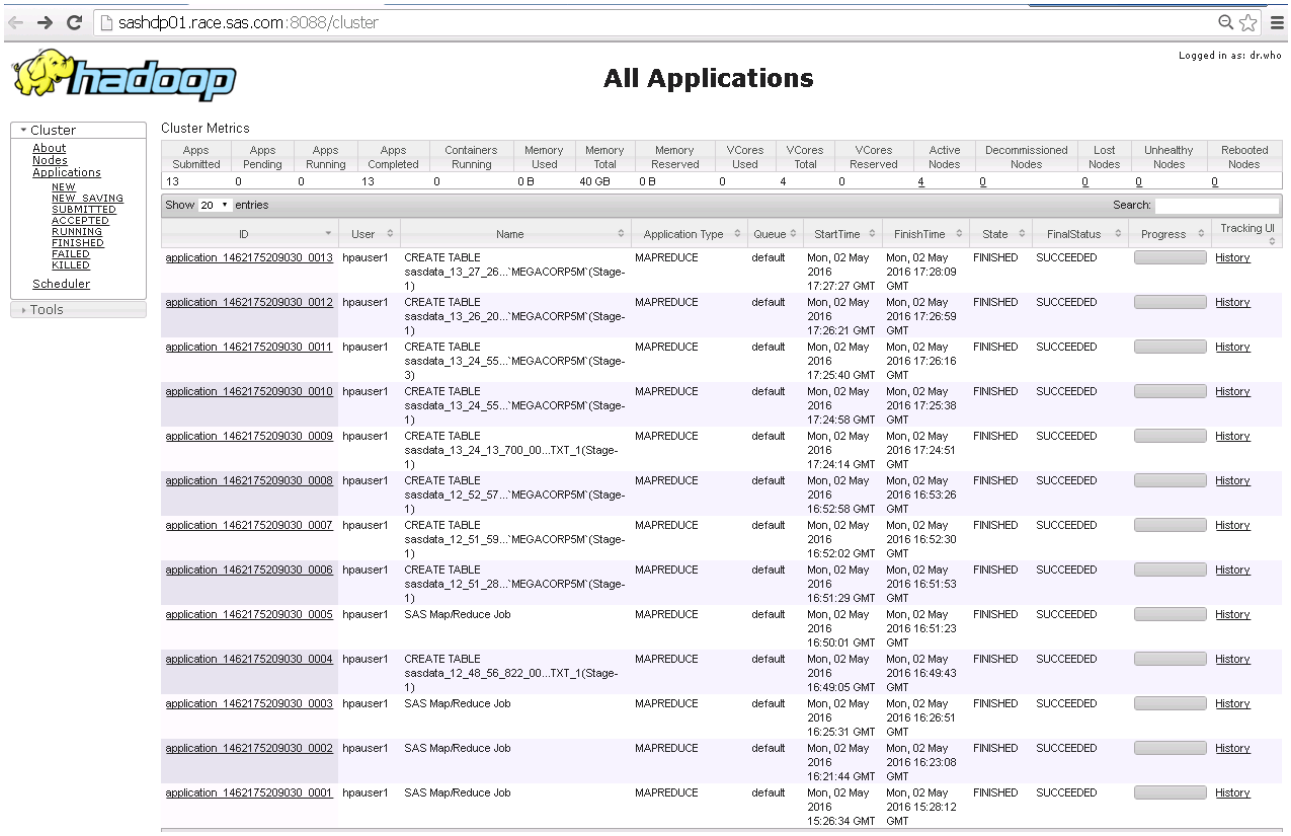
```
[root@sashdp01 ~]# yarn application -kill application_1449830114174_0003
15/12/11 12:56:57 INFO impl.TimelineClientImpl: Timeline service address:
http://sashdp01.race.sas.com:8188/ws/v1/timeline/
15/12/11 12:56:57 INFO client.RMProxy: Connecting to ResourceManager at
sashdp01.race.sas.com/10.96.3.101:8050
Killing application application_1449830114174_0003
15/12/11 12:56:59 INFO impl.YarnClientImpl: Killed application
application_1449830114174_0003
```

4.2.2 Resource Manager UI

- The Standard Resource manager web interface

The Resource Manager UI provides the same reports but via a web interface.

Standard URL: http://<Resource Manager Host>:8088/cluster

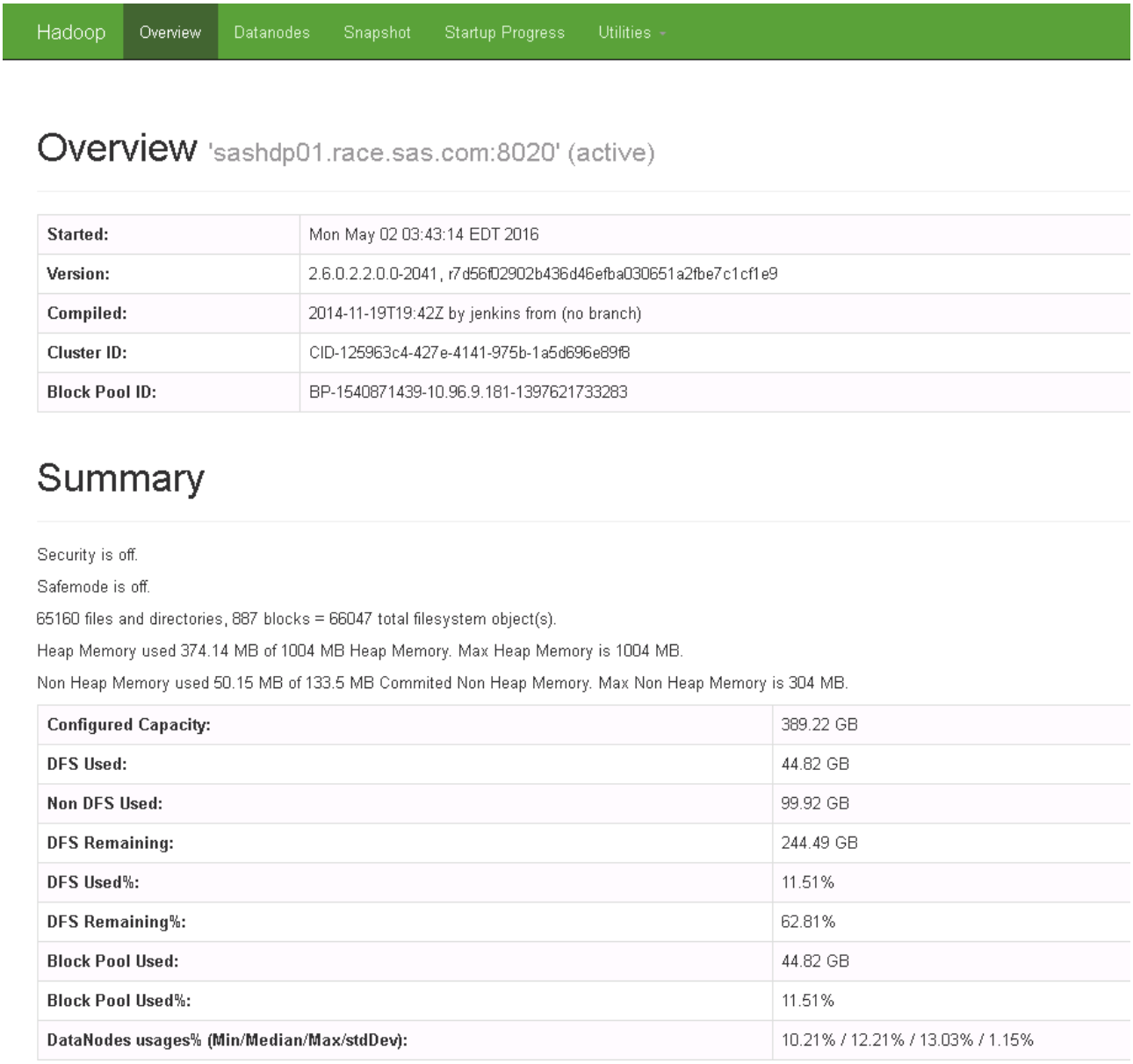


4.2.3 File System Shell and NameNode UI

To monitor your HDFS storage you can also use hadoop fs commands or the NameNode UI web console.

- NameNode UI

Standard URL: http://<name node host>:50070/



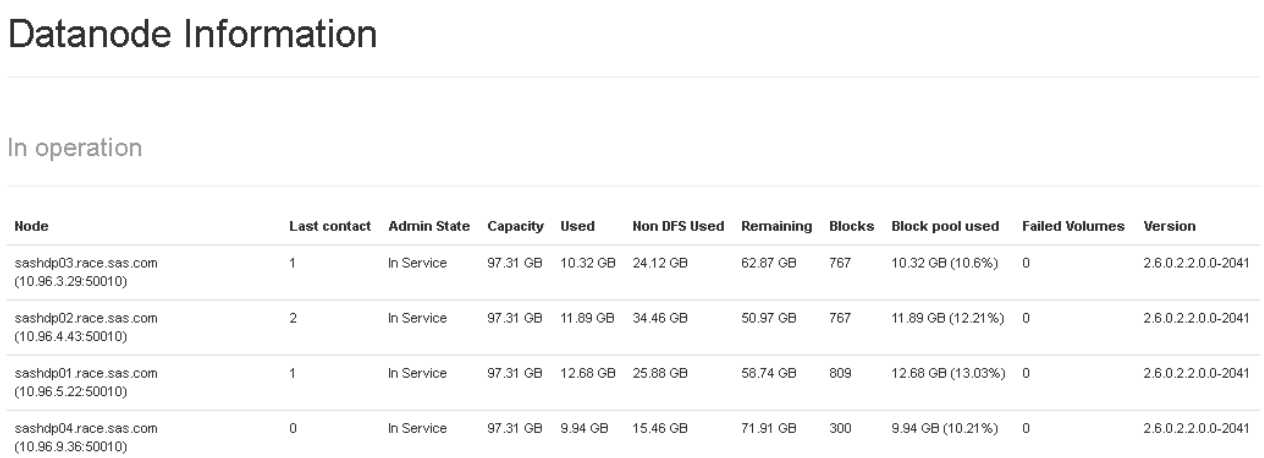
Configured Capacity gives the sum of the size of the file systems that are hosting data node directories (**dfs.datanode.data.dir**).

*Note: The **dfs.datanode.data.dir** property specifies the location on the local file systems where the HDFS blocks will be stored.*

DFS Used gives the accumulated size of HDFS data across all the nodes (corresponds to the sum of disk usage of the dfs.datanode.data.dir folder on all data nodes).

Non-DFS Used gives the accumulated size of disks used on the dfs.datanode.data.dir file system but outside the dfs.datanode.data.dir folder.

The **Datanodes** tab allows you to have a view of the remaining space on all data nodes.



Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
sashdp03.race.sas.com (10.96.3.29:50010)	1	In Service	97.31 GB	10.32 GB	24.12 GB	62.87 GB	767	10.32 GB (10.6%)	0	2.6.0.2.2.0.0-2041
sashdp02.race.sas.com (10.96.4.43:50010)	2	In Service	97.31 GB	11.89 GB	34.46 GB	50.97 GB	767	11.89 GB (12.21%)	0	2.6.0.2.2.0.0-2041
sashdp01.race.sas.com (10.96.5.22:50010)	1	In Service	97.31 GB	12.68 GB	25.88 GB	58.74 GB	809	12.68 GB (13.03%)	0	2.6.0.2.2.0.0-2041
sashdp04.race.sas.com (10.96.9.36:50010)	0	In Service	97.31 GB	9.94 GB	15.46 GB	71.91 GB	300	9.94 GB (10.21%)	0	2.6.0.2.2.0.0-2041

The remaining values in the table correspond to the available space in the file systems corresponding to the dfs.datanode.data.dir property.
The sum corresponds to the DFS remaining space in the Summary view.

- Filesystem shell commands

The UNIX equivalent of the du (Disk Usage) command is available in HDFS and can be very useful to detect directories taking a lot of space (https://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-common/FileSystemShell.html#du)

Example:

```
[hdfs@sashdp01 ~]$ hadoop fs -du -h /
6.1 M    /app-logs
7.2 G    /apps
438.5 M  /hdp
0        /hpatests
16.0 M   /hps
0        /mapred
4.3 M    /mr-history
2.5 K    /sas
0        /system
0        /test
1.3 G    /tmp
5.8 G    /user
0        /vapublic
```

4.3 Other monitoring tools (Ambari, CM, Hue)

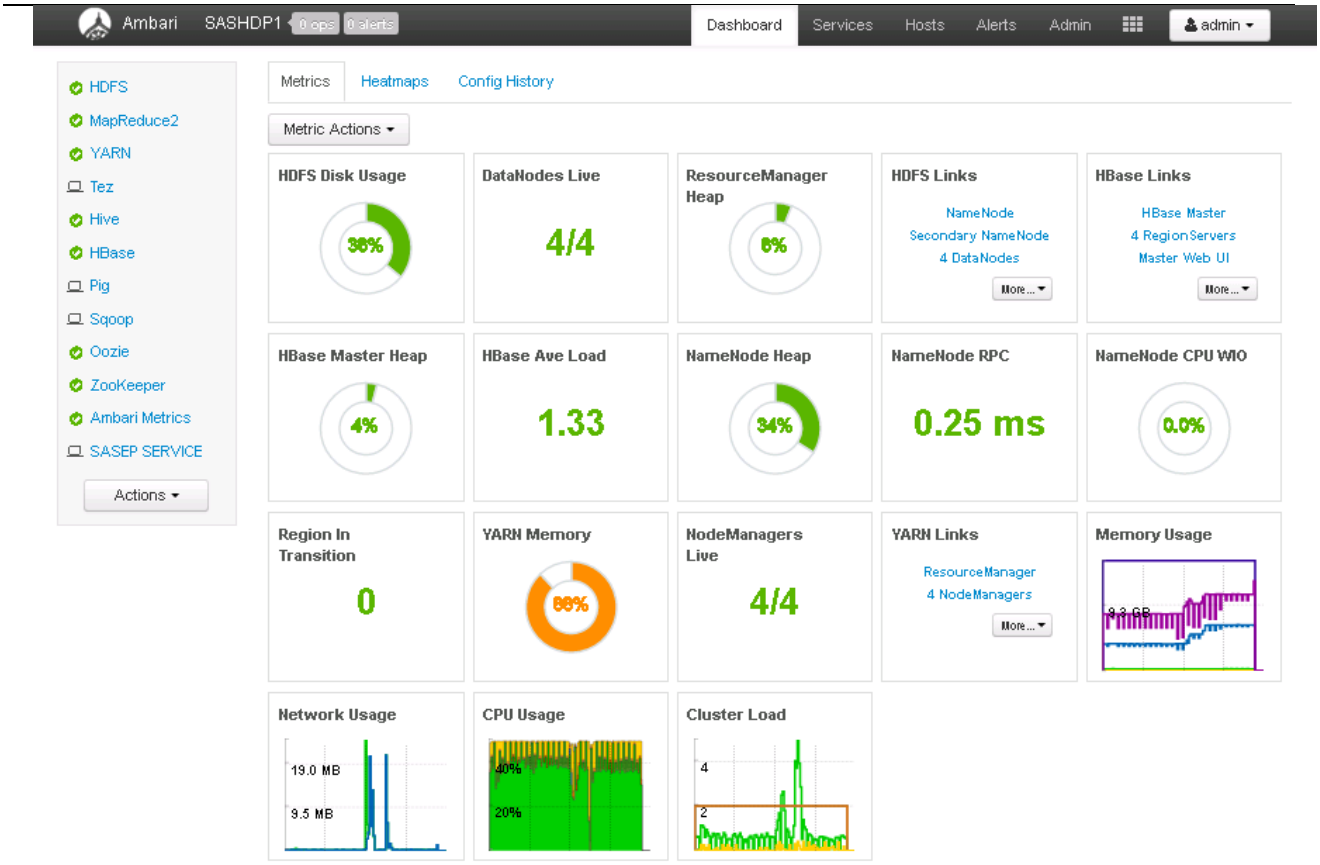
We will provide some examples of the monitoring capabilities for the 2 most common Hadoop administration tools: Ambari, Cloudera Manager and Hue.

4.3.1 Ambari console

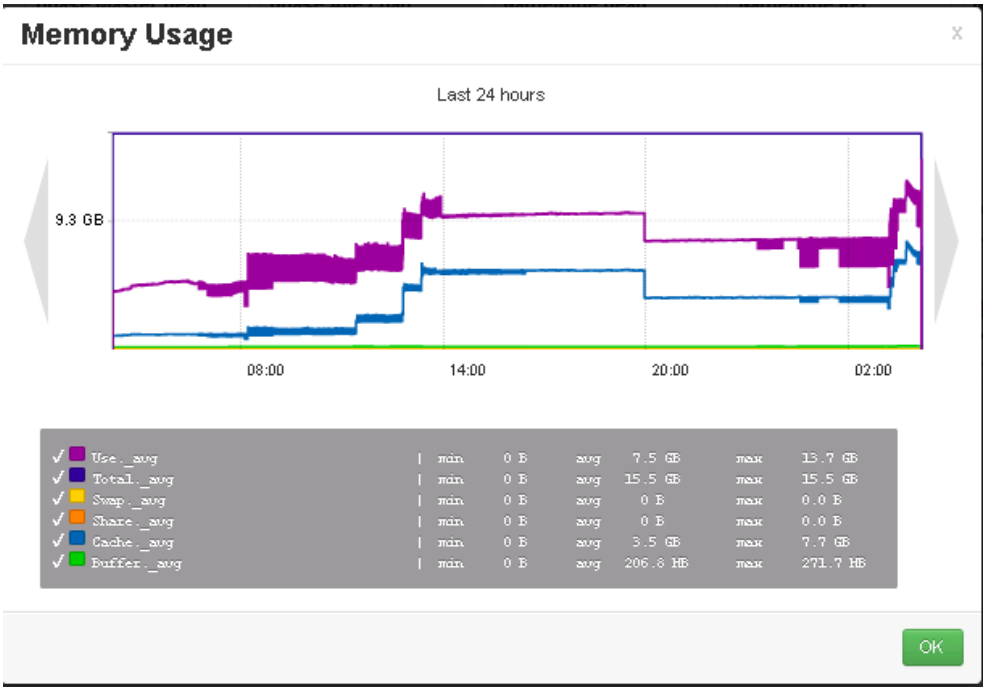
Ambari is an open source project providing a management console for Hadoop clusters. Ambari is used by default by several distributions such as Hortonworks and MapR.

- Ambari Metrics

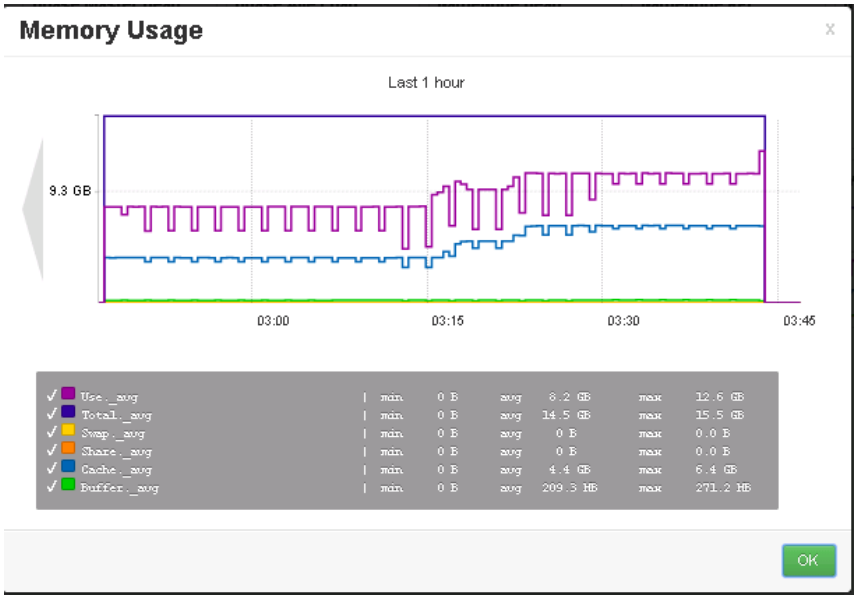
  In recent Ambari releases, Ganglia dashboards have been replaced by Ambari Metrics dashboards.



As we can see, metrics provide real-time information on the remaining space in HDFS and on YARN memory usage, for example.

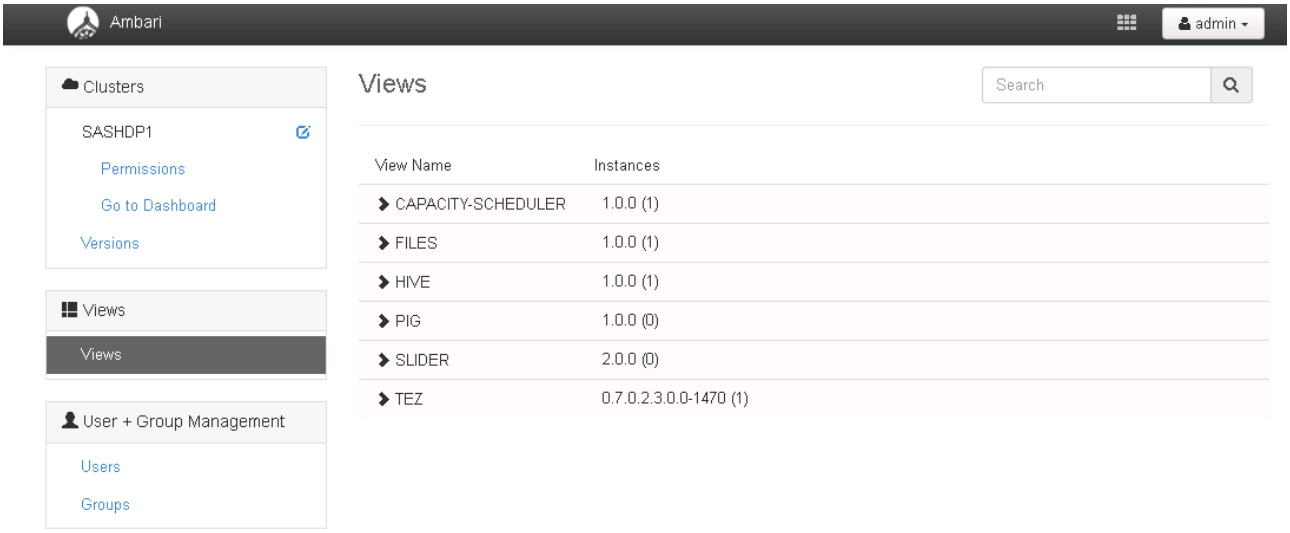


You can also zoom in on the metrics and consult the history.



- Views

In recent versions of Ambari, the concept of views extends Ambari capabilities and allows you to define additional UI to monitor specific services or provide additional features on specific components of the cluster.

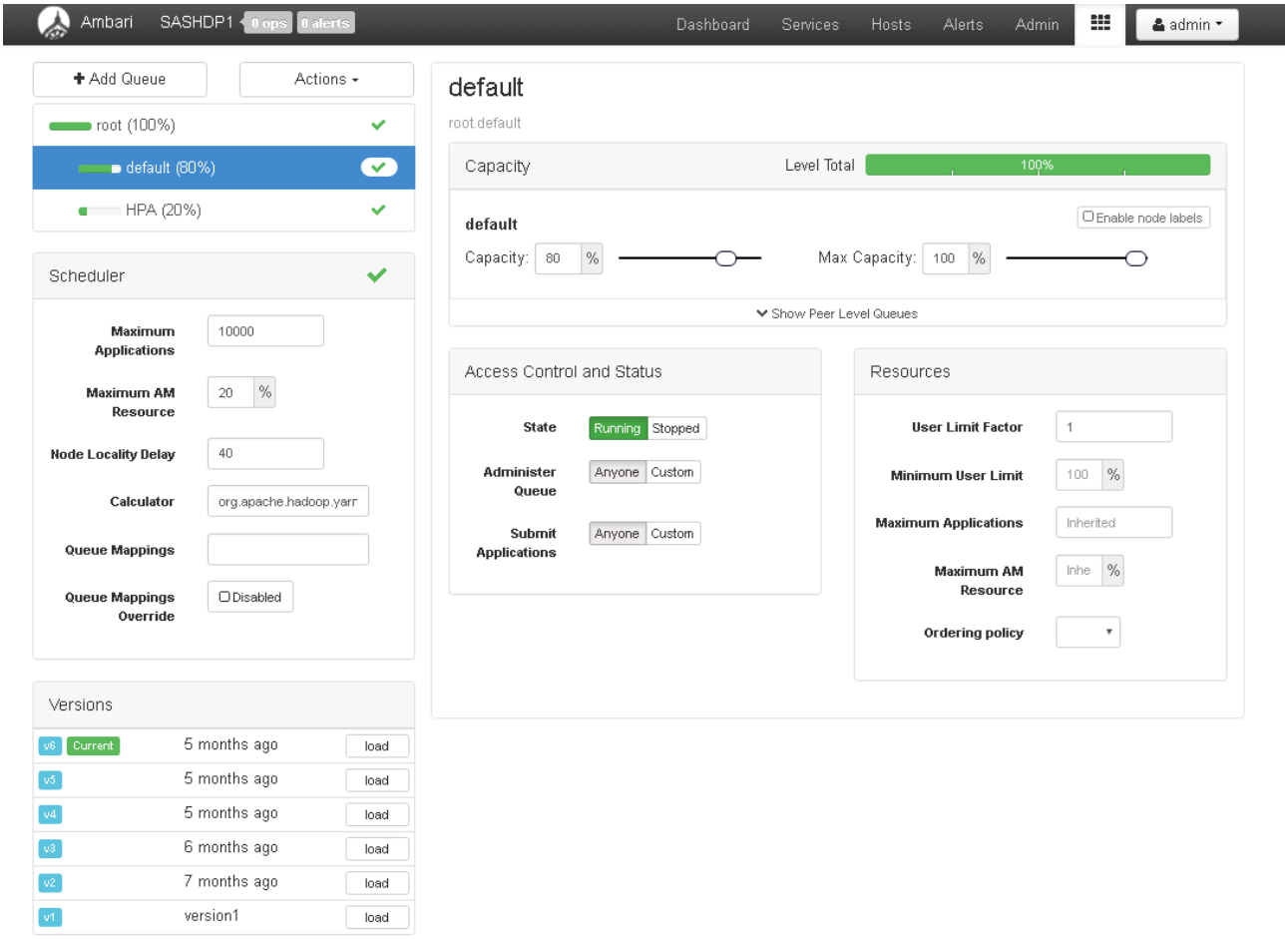


For example, in the Babar Collection (a RACE collection that is not publicly available), the following views were defined for:

- Capacity Scheduler
- Hive
- Tez

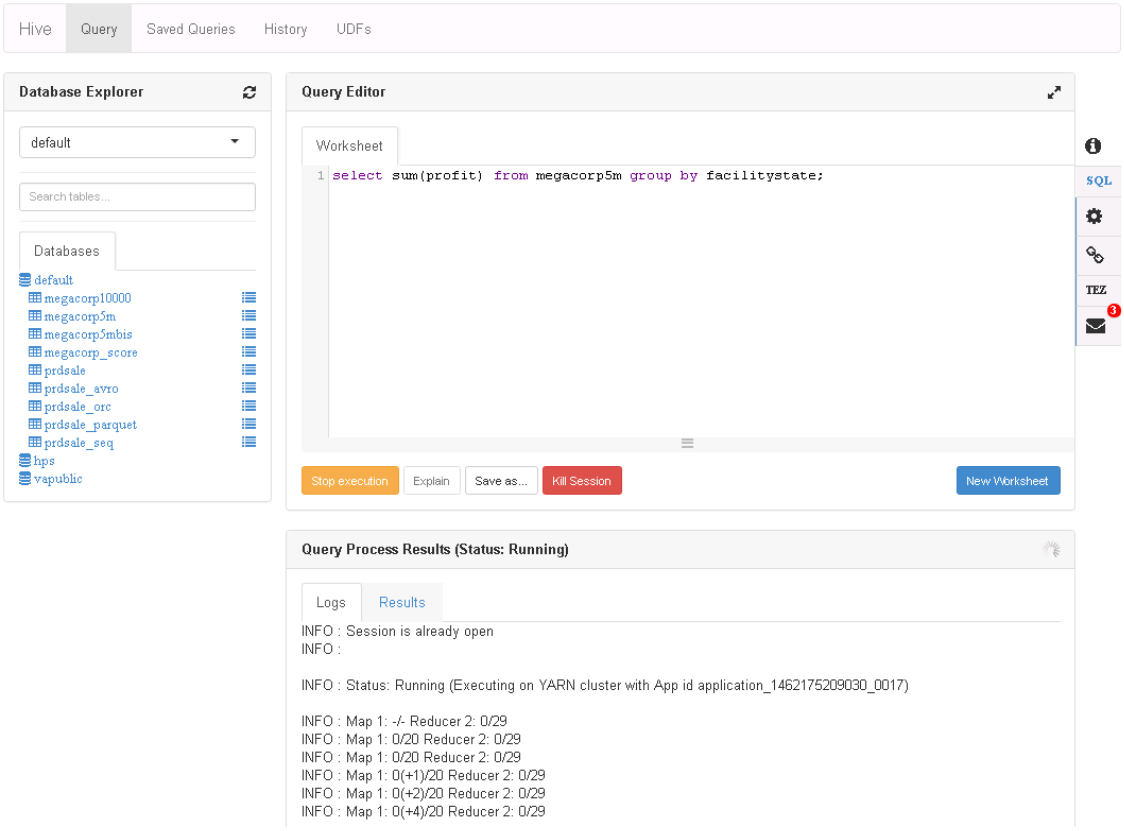
1. Capacity Scheduler view

This view allows you to configure queues using the YARN Capacity Scheduler.

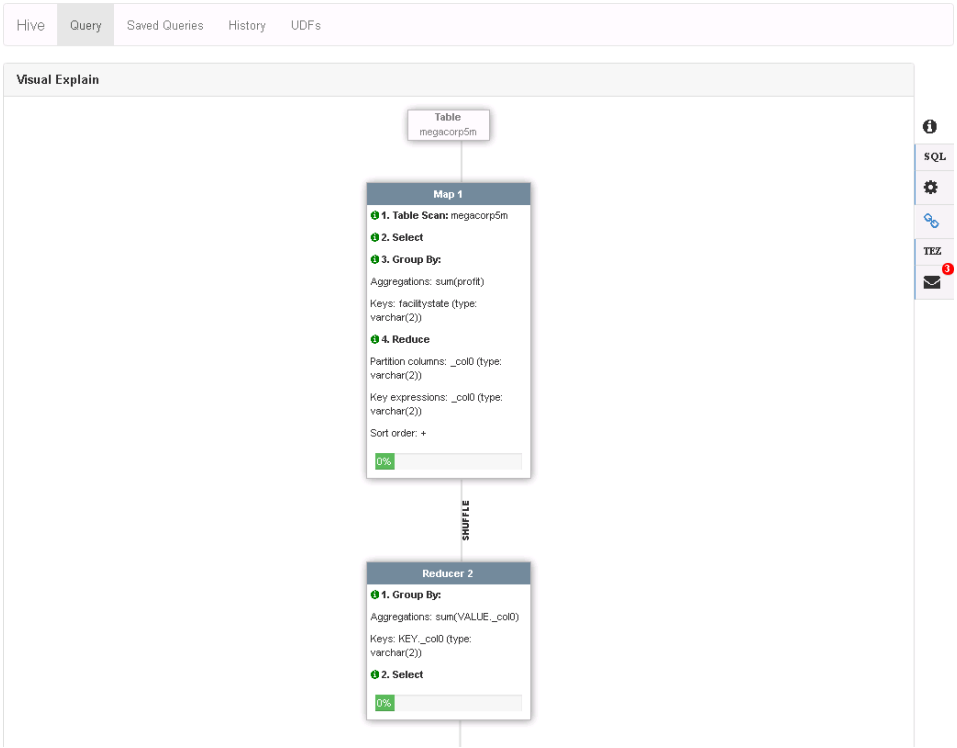


2. Hive view

This view provides a query editor, with a real-time monitoring window.

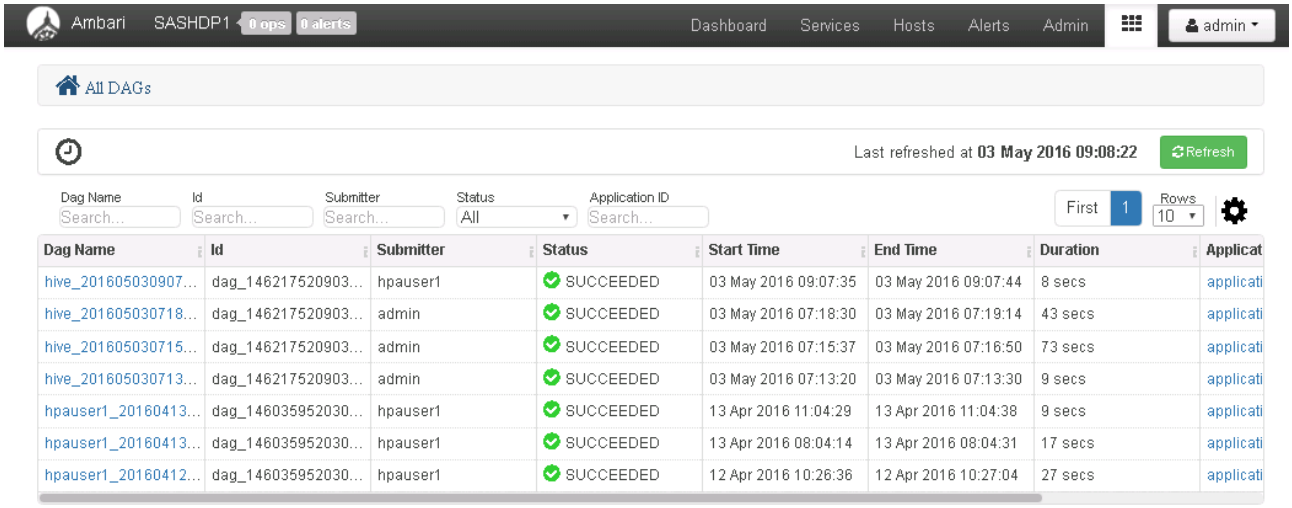


You can also have a visual representation of your query.



3. Tez view

This view allows you to monitor job execution when the engine is Tez.

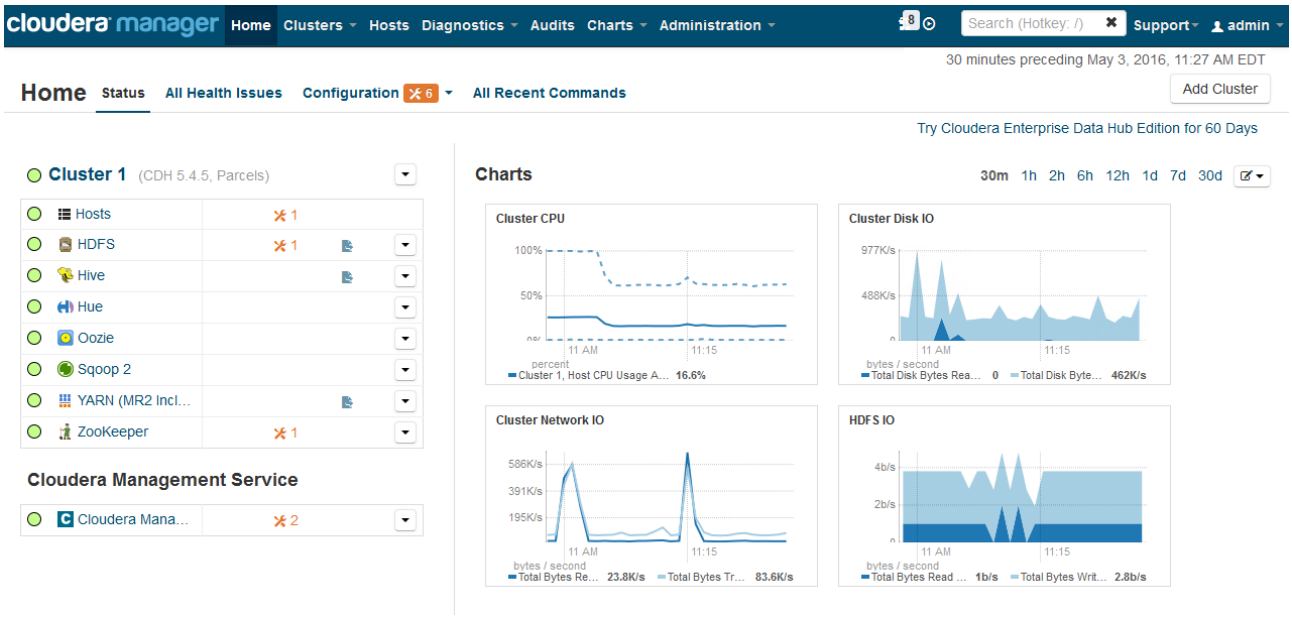


It provides details on jobs and the associated directed acyclic graph (DAG) representation.

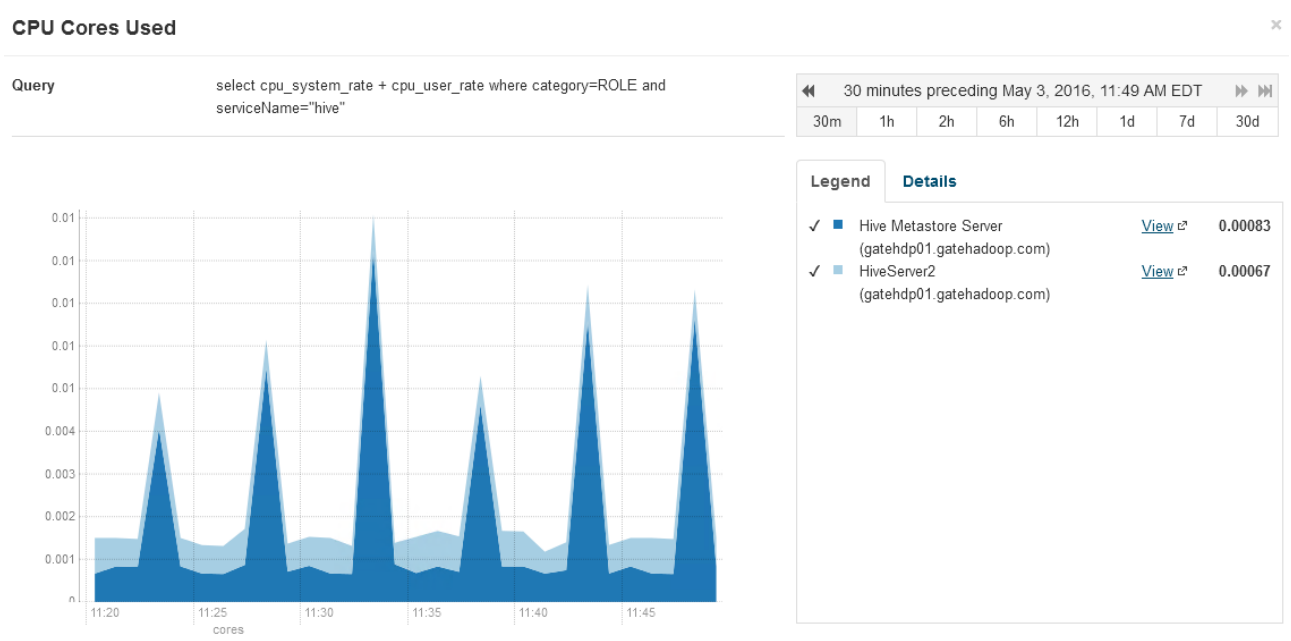
4.3.2 Cloudera Manager

- Cluster dashboards (Charts)

Cloudera provides its own Cluster dashboards:

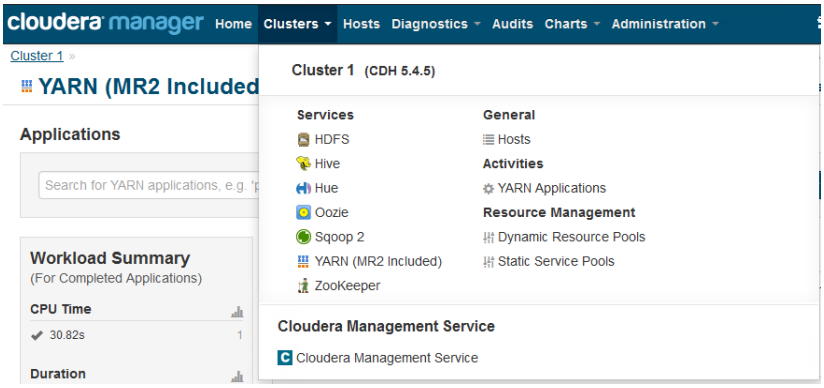


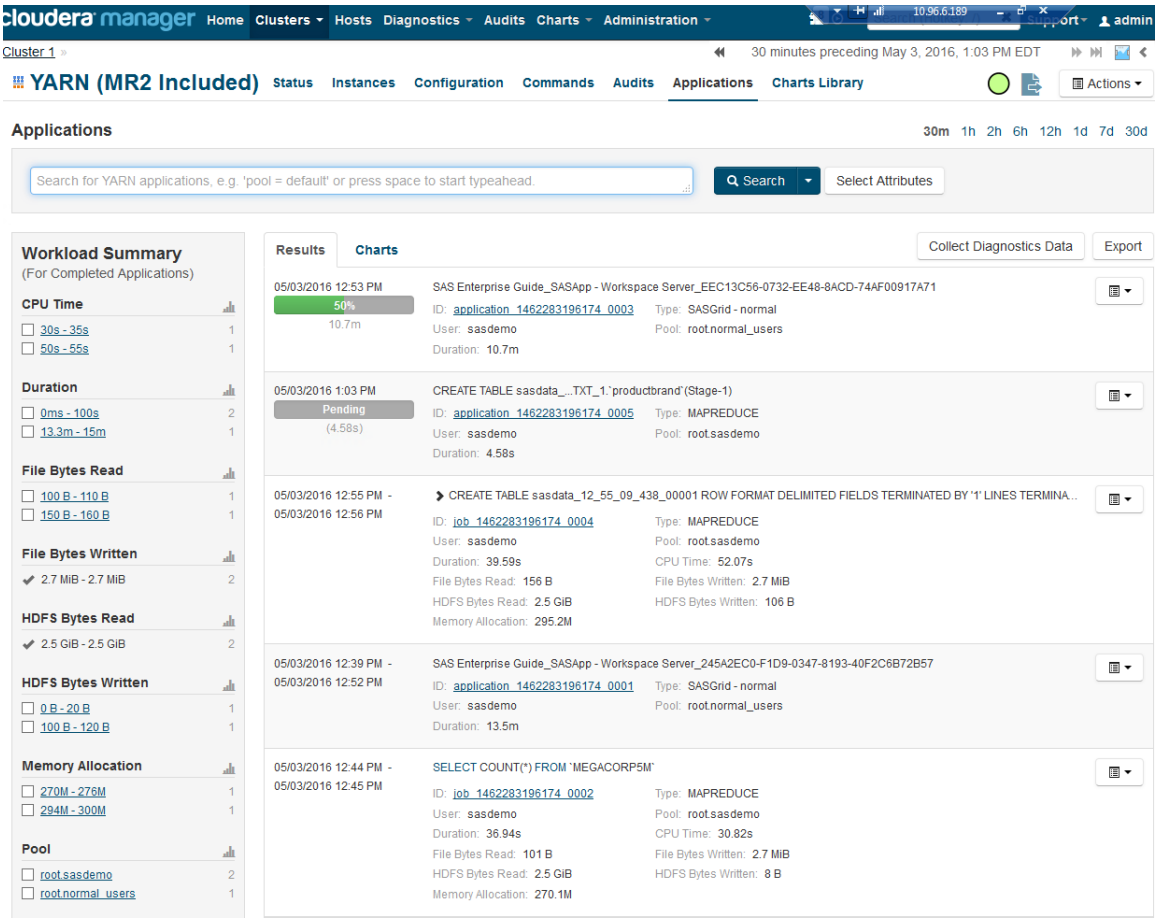
Depending on the service that you choose, you will have specific charts, and you can also zoom in and scroll back in the history.



- YARN applications

To display the monitoring UI, click **Clusters**, then **YARN Applications**.



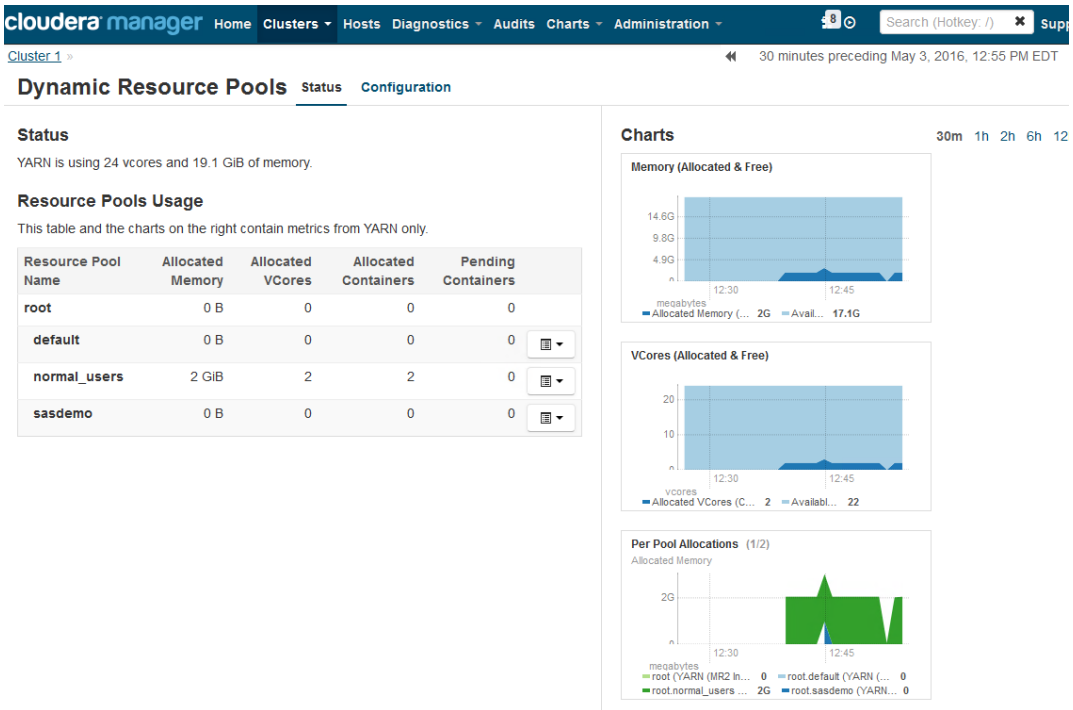


In the example above, we can see running or completed SAS Grid jobs and MapReduce jobs.

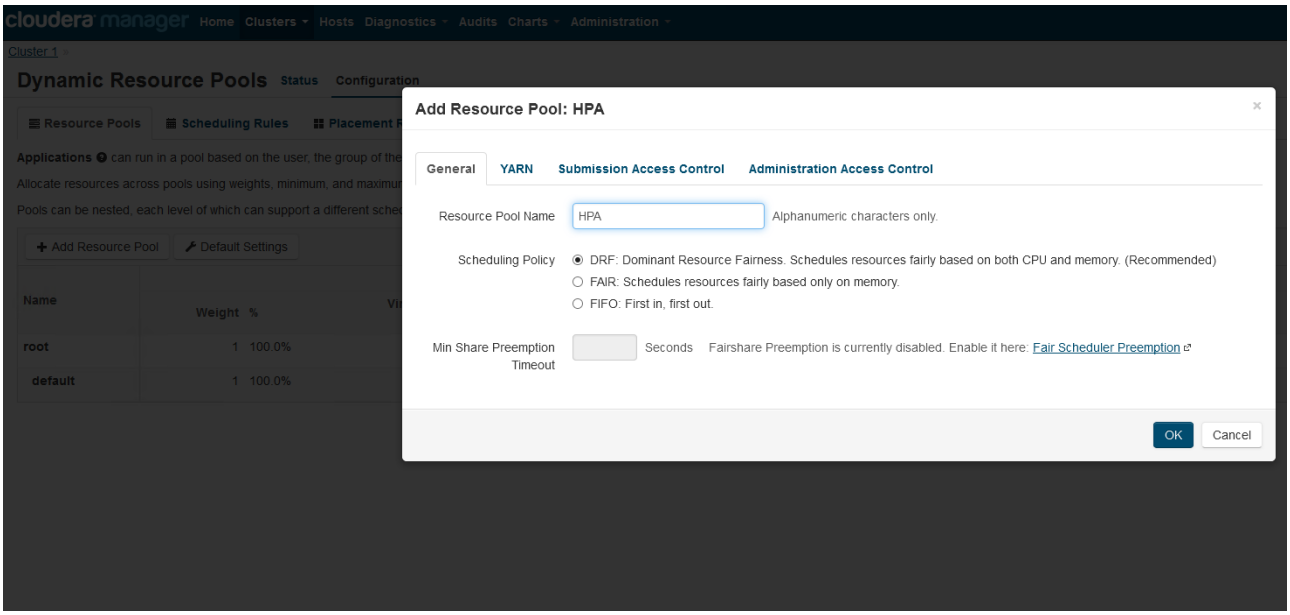
One nice feature is that we can see the complete SQL code (see example below for a PROC FREQ):

- Dynamic resource pools

In the example below we can see that two queues (or dynamic resource pools), normal_users and sasdemo, were dynamically created to manage SAS Grid jobs and MapReduce jobs, respectively.



It is also in this screen that you can define YARN queues to manage the cluster workload.



Note: Hue is generally included in Cloudera and integrated in the Cloudera Manager UI.

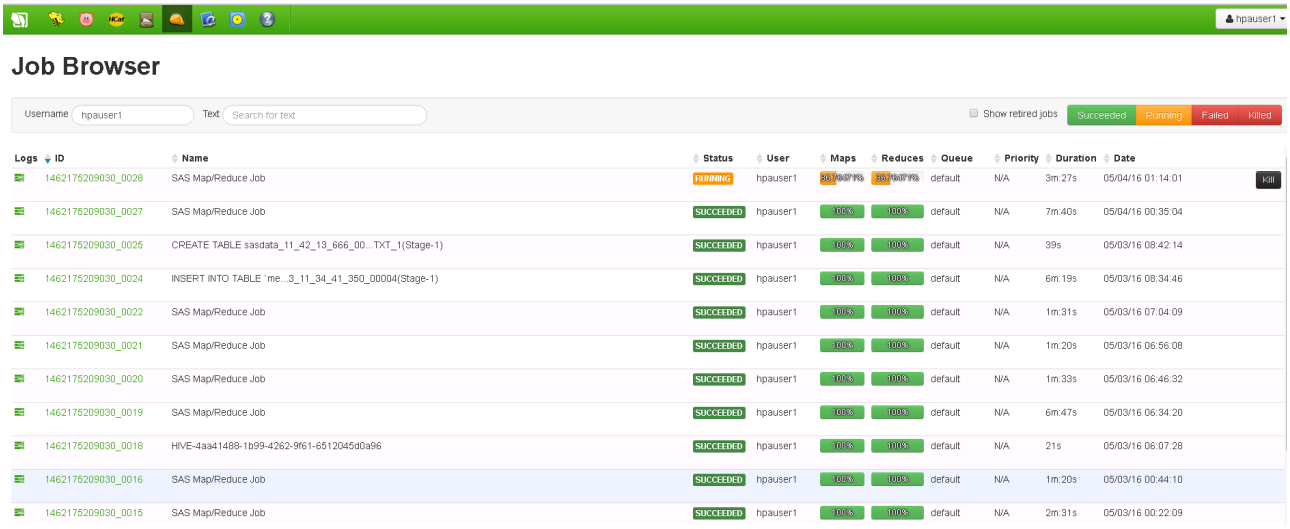
4.3.3 Hue

From the Hue User Guide:

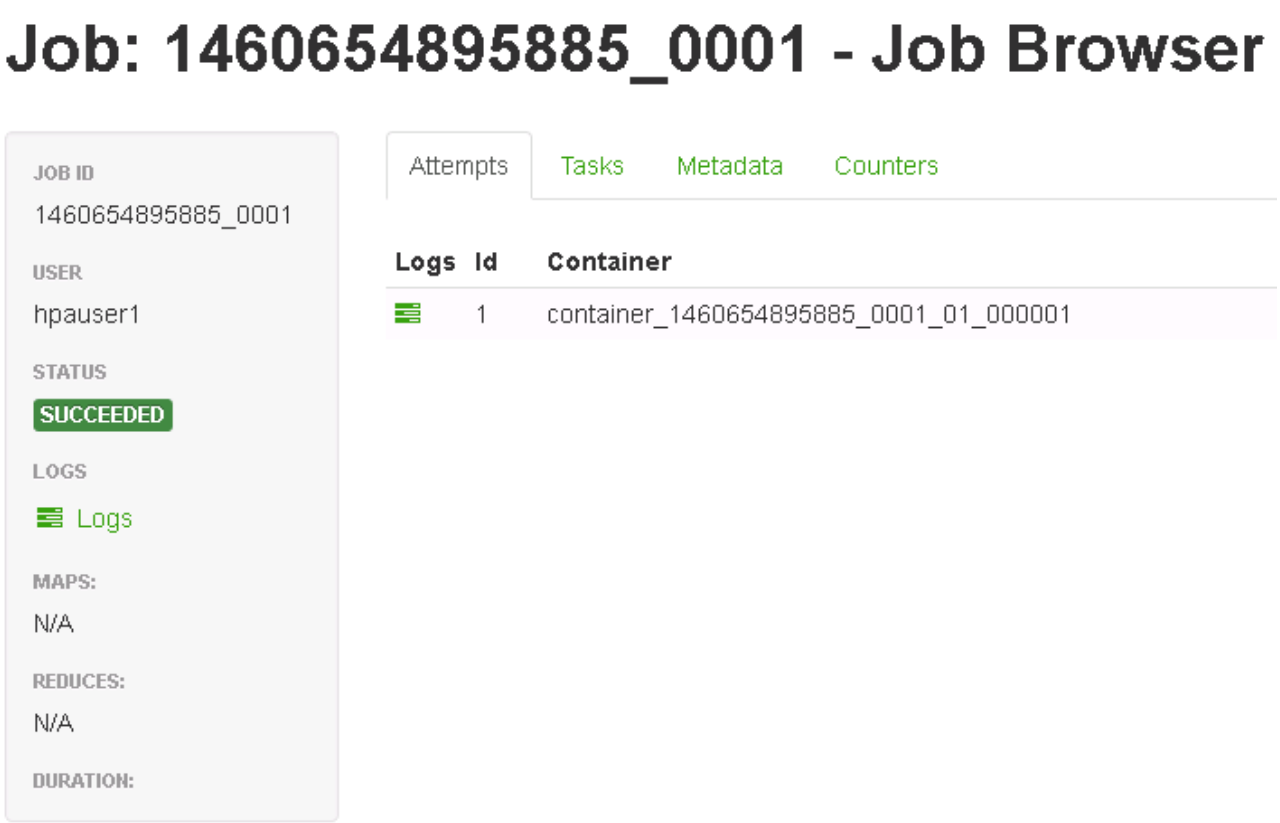
Hue is a browser-based environment that enables you to interact with a Hadoop cluster. Hue includes several easy to use applications that help you work with Hadoop MapReduce jobs, Hive queries, Hadoop files and user accounts. The Hue applications run in a Web browser and require no client installation.

Standard URL: http://<Hue Server Host>:8000/jobbrowser/

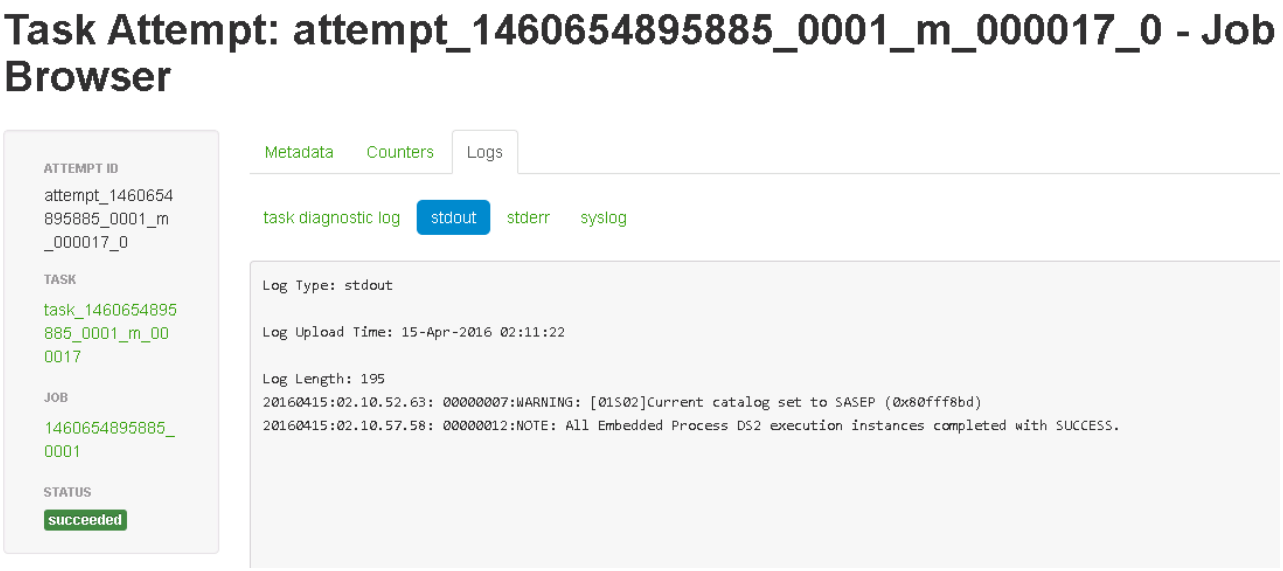
You have a nice interface to follow the job execution:



Then once the job is finished, it is very easy to see the job tasks and statistics:



The logs are easily readable from this interface, too:



Note that Hue also provides the following features:

- An advanced HDFS browser (allowing Read and Write operations in HDFS)
- Beeswax: a Hive query editor (with queries stats, execution plan, and so on)

4.4 Monitor ZooKeeper connections

ZooKeeper is used by HiveServer to manage concurrent transaction (required by the Hive's Table Lock Manager).

When a very high number of Hive queries are submitted at the same time by a client application (for example, in SAS Data Integration Studio), it can happen that the number of open ZooKeeper connections exceeds the maximum number of connections defined in the default configuration. In such cases, SAS jobs might fail, as they cannot open a new Hive connection.

5 Appendix

5.1 Limitations of the testing environment used here

The findings presented here are not representative of exhaustive testing. This is a research document, designed to stimulate further work.

The test supporting the content of this document was done under the following conditions:

- Third maintenance release of SAS 9.4 and the SAS Embedded Process 9.43
- Hortonworks Data Platform 2.2.0 deployed on 4 Linux Servers (Red Hat Enterprise Linux 6) acting as data nodes. Each server has 16GB RAM with Intel® Xeon® CPU X7560 @ 2.27GHz (2 cores).

The size of the table used in all tests is 8.6 gigabytes (small in the Hadoop world). The table has 30 million rows and 48 variables (37 are numeric). The Hadoop environment (Babar collection, not publicly available) is small and not a candidate to run official benchmarks on. In the real world, Hadoop clusters are much bigger. Customers can have 100 to over 1,000 nodes in their Hadoop environments.

SAS and Hadoop are always evolving so the system behavior may change over time.

For all these reasons the results presented here are mainly informative.

5.2 Hadoop file type benchmark

For real benchmark results, ask your SAS representative who can work with SAS Enterprise Excellence Center (EEC). Several benchmarks have been done with data stored in Hadoop and can be made available on request.

The purpose of the following table is simply to show that depending on the operations, the choice of the Hadoop file format can impact the performance.

	MEGACORP 30m rows					
	HIVE table (text/csv file)	SPDE Table	Colum-oriented file in HIVE (ORC)	SerDe file (AVRO)	HDMD (Direct HDFS)	SASHDAT
Copy from local (DS1)	real time 10:27.47 cpu time 6:17.25	real time 5:14.77 cpu time 20.76 seconds	real time 15:13.94 cpu time 3:33.82	real time 16:39.17 cpu time 4:36.37	real time 7:21.94 cpu time 3:32.88	real time 9:05.69 cpu time 26.95 seconds
PROC SQL (Count, max)	real time 11:10.97 cpu time 0.07 seconds	real time 1:32.16 cpu time 4.86 seconds	real time 43.74 seconds cpu time 0.06 seconds	real time 6:04.40 cpu time 0.10 seconds	real time 38:00.47 cpu time 35:40.04	NA (Not supported with SASHDAT)
PROC HPSUMMARY	real time 12:00.29 cpu time 4.33 seconds	real time 4:03.67 cpu time 7.01 seconds	real time 10:25.94 cpu time 7.34 seconds	real time 12:05.22 cpu time 6.72 seconds	real time 6:15.97 cpu time 4.35 seconds	real time 2:50.40 cpu time 6.86 seconds
PROC HPLOGISIC	real time 15:34.51 cpu time 4.44 seconds	real time 6:08.51 cpu time 6.07 seconds	real time 7:29.77 cpu time 4.62 seconds	real time 12:47.74 cpu time 7.19 seconds	real time 7:06.19 cpu time 4.44 seconds	real time 1:26.96 cpu time 4.23 seconds
PROC HPGENSELECT	real time 17:55.71 cpu time 7.41 seconds	real time 6:24.60 cpu time 10.17 seconds	real time 13:04.90 cpu time 6.65 seconds	real time 16:36.79 cpu time 15.01 seconds	real time 11:42.27 cpu time 7.45 seconds	real time 3:55.72 cpu time 8.23 seconds

5.3 References

Official guides

- SAS® 9.4 Hadoop Configuration Guide for Base SAS® and SAS/ACCESS®
  http://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoopbacg.pdf
- SAS/ACCESS® 9.4 for Relational Databases: Reference
  http://support.sas.com/documentation/cdl/en/acreldb/69039/PDF/default/acreldb.pdf
- SAS® 9.4 In-Database Products: User's Guide
  http://supportprod.unx.sas.com/documentation/cdl/en/indbug/68442/PDF/default/indbug.pdf
- SAS® 9.4 DS2 Language: Reference
  http://support.sas.com/documentation/cdl/en/ds2ref/68052/PDF/default/ds2ref.pdf

SPDE format in HDFS

- SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System
  http://support.sas.com/documentation/cdl/en/engspdehdfsug/67948/PDF/default/engspdehdfsug.pdf
- The SAS® Scalable Performance Data Engine: Moving Your Data to Hadoop without Giving Up the SAS Features You Depend On
  http://support.sas.com/resources/papers/proceedings15/SAS1956-2015.pdf
- SAS® SPD Engine and Hadoop Working Together: Requirements and Best Practices
  http://support.sas.com/resources/papers/SPDE_Hadoop.pdf

Other useful papers or links

- Deploying SAS® High Performance Analytics (HPA) and Visual Analytics on the Oracle Big Data Appliance and Oracle Exadata
  http://www.oracle.com/technetwork/database/bi-datawarehousing/sas/sas-hpa-va-bda-exadata-2389280.pdf
- Best Practices for YARN Resource Management
  https://www.mapr.com/blog/best-practices-yarn-resource-management

Global forum papers

- Exploring SAS® Embedded Process Technologies on Hadoop®
  http://support.sas.com/resources/papers/proceedings16/SAS5060-2016.pdf
- Best Practices for Resource Management in Hadoop
  http://support.sas.com/resources/papers/proceedings16/SAS2140-2016.pdf
- Leveraging Big Data Using SAS® High-Performance Analytics Server
  http://support.sas.com/resources/papers/proceedings13/399-2013.pdf

SAS INSTITUTE INC. WORLD HEADQUARTERS SAS CAMPUS DRIVE CARY, NC 27513
TEL: 919 677 8000 FAX: 919 677 4444 U.S. SALES: 800 727 0025 **WWW.SAS.COM**