Paper SAS6520-2016

SAS® and Hadoop: The 5th Annual State of the Union

Paul Kent, SAS Institute Inc.

ABSTRACT

The third maintenance release of SAS® 9.4 (aka SAS 94m3) was a huge release with respect to the interoperability between SAS® and Hadoop -- the industry standard for big data. This talk brings you up-to-date with where we are: more distributions, more data types, more options. Come and learn about the exciting new developments for blending your SAS processing with your shared Hadoop cluster. Grid processing. Check. SAS data sets on HDFS. Check

INTRODUCTION

Hadoop and the era of using several computers at once to solve your problem promised to change analytics back in 2010, if not 2006 (Hadoop has just celebrated its 10th birthday). Similar waves of change have had a significant change on SAS Software, and this time around is no different.

This paper will introduce some basic Hadoop concepts, and describe the ways a SAS programmer can interact with data stored in a Hadoop File System. In fact 2016 is the first year I can recall where I didn't see a headline in the run-up to SAS Global Forum asking "Are YOU Hadooping yet?"

After a short introduction to Hadoop, I'll detail several ways a SAS programmer interfaces to Hadoop. At the very most basic level, it is just another source of raw data. Hadoop has a flexible approach towards defining a tabular shape for data -- hive, and this is the next tier of integration with SAS. Finally there are SAS approaches for full-on adoption of Hadoop divide and conquer principles of operation.

In the talk I liken these stages to those of human courtship. At first the attraction might be driven by curiosity, introductions are made and before long the parties are dating, become engaged, and possibly marry.

HADOOP INTRODUCED

Hadoop is a very successful (and visible) open source project, similar to Linux itself, and the Apache Web Server project which help establish the "Apache way" at the Apache Software Foundation – whose mission is "to provide software for the public good". From https://hadoop.apache.org …

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

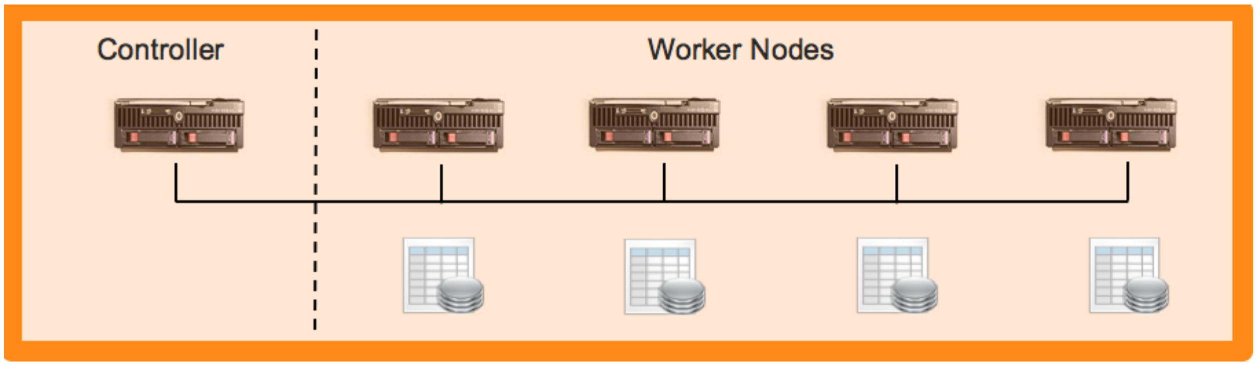
The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Yarn is a relative newcomer to the project. It is also the feature that allows enterprises to build large shared clusters that they can share widely over their constituent groups – Yarn makes it possible to administer access to a shared resource that is often times "over subscribed".

The essential idea behind Hadoop is to use a number of computers to solve the problem, and use them in such a way that you can divide and conquer the workload by sharing it across all the workers in the cluster. It is typical to have some form of management node that is the co-ordinator and controls the work distributed to the workers.



There are two beautiful ideas in the original Hadoop:

1. I will spread your data over many computers to keep it safe
2. I will facilitate a way to send the work to the data, instead of asking that you gather all the data together for processing

IDEA #1: HADOOP NEVER FORGETS

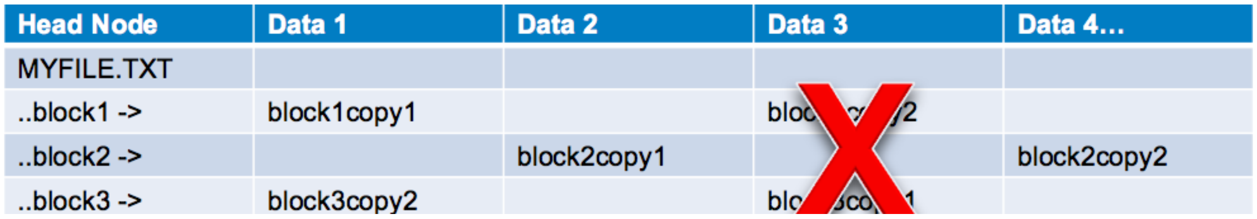
A file in HDFS is represented as blocks, and each block is sent to a different computer:

Head Node	Data 1	Data 2	Data 3	Data 4…
MYFILE.TXT				
..block1 ->	block1copy1			
..block2 ->		block2copy1		
..block3 ->			block3copy1	

This in and of itself doesn't increase the chances that your file is available – now you are at the mercy of 5 computers instead of just one (and usually a more reliable one). Hadoop compensates for this by storing redundant copies of the same blocks on different nodes in the cluster:

Head Node	Data 1	Data 2	Data 3	Data 4…
MYFILE.TXT				
..block1 ->	block1copy1		block1copy2	
..block2 ->		block2copy1		block2copy2
..block3 ->	block3copy2		block3copy1	

Even in this small cluster, you are now protected against a single data node failure. Suppose node "Data 3" develops a hardware fault… without redundancy you would not be able to locate "block3", but with even a single level of redundancy you can find the alternate copy of "block3" – on node "Data 1":



Head Node	Data 1	Data 2	Data 3	Data 4…
MYFILE.TXT				
..block1 ->	block1copy1		block1copy2	
..block2 ->		block2copy1		block2copy2
..block3 ->	block3copy2		block3copy1	

A Hadoop cluster is typically operated with "copies=2", or a redundancy factor of 3. This makes the files very resilient – they are available even in the face of several failures. Hadoop is proactive about keeping the availability high – 10 minutes after a block is missing (so there are only 2 instead of the required 3 copies), Hadoop will commission yet another copy of the block elsewhere in the cluster to maintain the "copies=" number of safe copies.

IDEA #2: SEND THE WORK TO THE DATA

Imagine you are at a wedding reception. To keep the numbers workable, the room contains 10 round tables, and there are 10 people at each table. The controller (Master of Ceremonies) is warming up the crowd and wants to find the youngest person in the room (in other words calculate the MIN statistic over the dataset of 100 records)

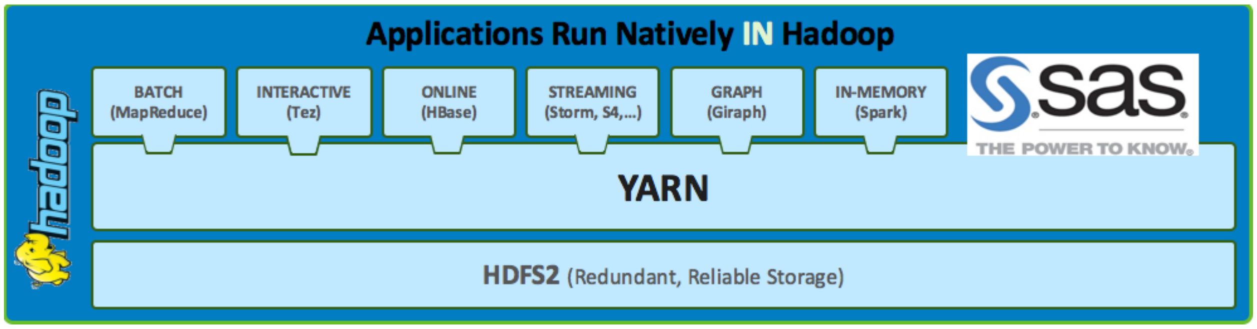
The default approach would have the MC visit each table in turn, ask each person their age, and keep a running memory of the youngest person thus far – A sequential scan of the dataset.

The Hadoop approach engages the crowd. The MC asks the youngest person at each table to come to the center of the room, where a quick election can be made as to the youngest of the youngest. This is distributed, or parallel computing, and it does not rely on one computer making a sequential scan of the entire dataset. In fact the problem (or solution rather) scales nicely as the wedding grows – either add 5 more tables (grow the cluster), or add 5 people to each table (grow the capacity of each node)

Idea#2 isn't very useful without Idea#1 having spread the data out over many computers to start with.

IDEA #3: MAKE IT EAST TO SHARE THE RESOURCES OF THE CLUSTER

These first two ideas were sufficient to get Hadoop adopted by the early "Web Scale" computing centers like Google, Yahoo, Facebook and LinkedIn. The 3rd idea that we needed a good way to share the resources of the cluster resulted in traditional enterprises being interested too – the cost dynamics of Hadoop were one- (and sometimes two-) order of magnitude better than the previous approach.



YARN is the component of Hadoop that interposes itself between the distributed, reliable storage subsystem (HDFS), and the applications that want to process that data. Applications make resource requests to the YARN controller, who allocates them out, possibly after waiting for said resource to become available again after a different application is finished using it.

SAS AND HADOOP – GETTING CONNECTED

The SAS FILEREF has traditionally been used to refer to external data – initially to reference one of the DD cards in the JCL of the Job you submitted to your mainframe, but more recently a fileref has been seen referring to a path on the operating system, to a directory of a FTP server, and to a page (URL) of a website. The SAS fileref has been extended to allow you to specify a file in HDFS

```
FILENAME paul HADOOP
   '/users/kent/mybigfile.txt'
   CONFIG='/etc/hadoop.cfg' USER='kent' PASS='sekrit';

DATA MYFILE;
   INFILE paul;
   INPUT name $ age sex $ height weight;
   RUN;
```

Many details regarding the Hadoop cluster you want to connect to are not repeated in each and every program – you simply refer to a configuration file for the cluster. Your system admin folks will have set this file up for you, so ask around until you find a colleague with one that works and copy his/her approach ☺

At this point your SAS program can read or write data to a Hadoop cluster. You are not yet enjoying all the possible benefits Hadoop could bring – especially, you are still copy the data between the SAS Server and the Hadoop cluster. This might be just the thing for an ETL job where you are adding the next hours worth of web-site-visits to the collection, but you certainly don't want to copy the past 10 years web-site-visits back out of the cluster to do some analytics!

The HADOOP fileref allows your SAS program to connect with data in Hadoop. PROC HADOOP is the counterpart that allows you to interface with the control of the Hadoop cluster.

```
PROC HADOOP CFG=CFG … [VERBOSE];
   HDFS <hdfs commands>;
   MAPREDUCE <mapreduce options>;
   PIG <pig options>;
   RUN;

 /* file and directory manipulation */
   hdfs mkdir='/tmp/rick/mydir';
   hdfs copyfromlocal='myfile.txt' out='/tmp/rick/mydir/myfile.txt';
   hdfs copytolocal='/tmp/rick/mydir/myfile.txt' out='myfile2.txt';
   hdfs delete='/tmp/rick/mydir/myfile.txt';

 /* new with SAS 94m3 */
   hdfs ls='/tmp/rick';
   hdfs ls='/tmp/rick' out=lsfile;
   hdfs cat='/tmp/rick/testfile.txt';
   hdfs cat='/tmp/rick/*.txt out=catfile;
```

SAS AND HADOOP – DATING

Relational Databases like Oracle, DB2, Teradata and SQL Server, and to a large extent SAS itself have spoiled us to expect more from our data sources than the typical CSV (comma separated values) file. We have grown accustomed to a richer description of the data elements in the collection – the variable names, the data type (is it a date?) and even formatting niceties like a Label and DisplayFormat.

We process these tables with Languages like the SAS DATA Step, Standards Based SQL and Open Source concepts like the data frame. And we expect the table metadata to be durable – that the information it represents gets transferred into the language we are using.

HIVE is the Hadoop subsystem responsible for managing and presenting tabular style metadata about files stored in HDFS, and SAS® Access for Hadoop is the SAS Software that interfaces to HIVE so that these Hadoop tables appear with High Fidelity as SAS tables in your SAS session. HIVE operates using well known SQL principles. There are a pair of Hadoop specific vendor implementations of SQL and corresponding SAS Access products to interoperate with them – SAS® ACCESS to Impala and SAS® ACCESS to Hawq

You use the SAS Libname statement to reference the collection of tables in HIVE

```
LIBNAME olly HADOOP
   SERVER=olly.mycompany.com
   USER='kent' PASS='sekrit';

PROC DATASETS LIB=OLLY;
   RUN;
```

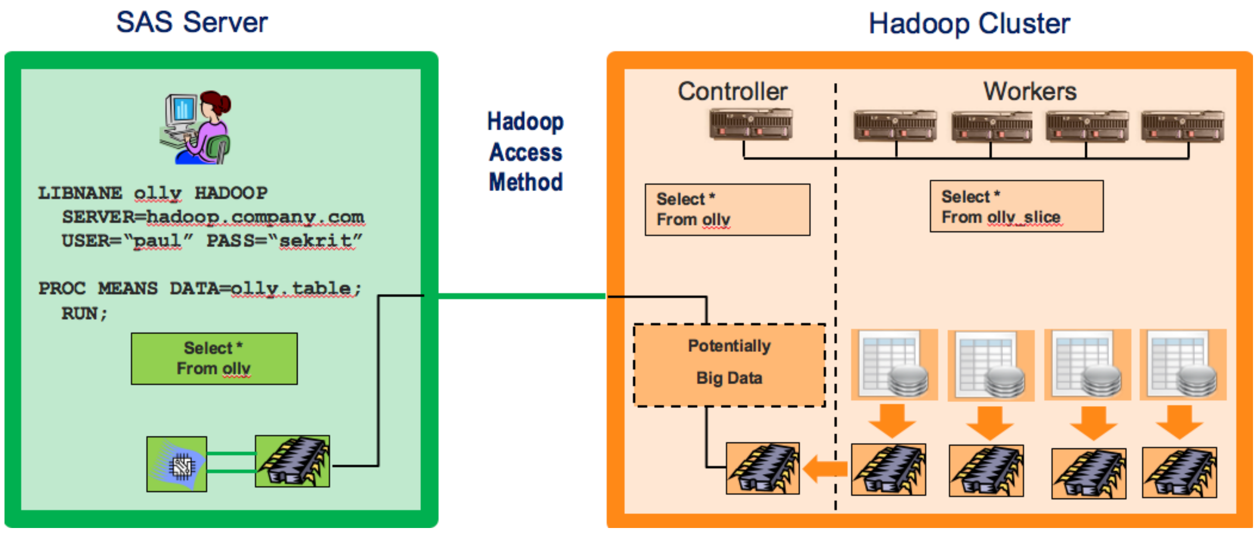
One of the reasons I refer to this as dating is that more information is exchanged across the boundary between SAS and Hadoop – all that rich metadata regarding the contents of the files. This is also the stage where we can see some of the work being transferred into the cluster rather than the rows of data being extracted over to the SAS server.

SQL PASSTHRU – ASK THE DATA SOURCE TO DO THE WORK

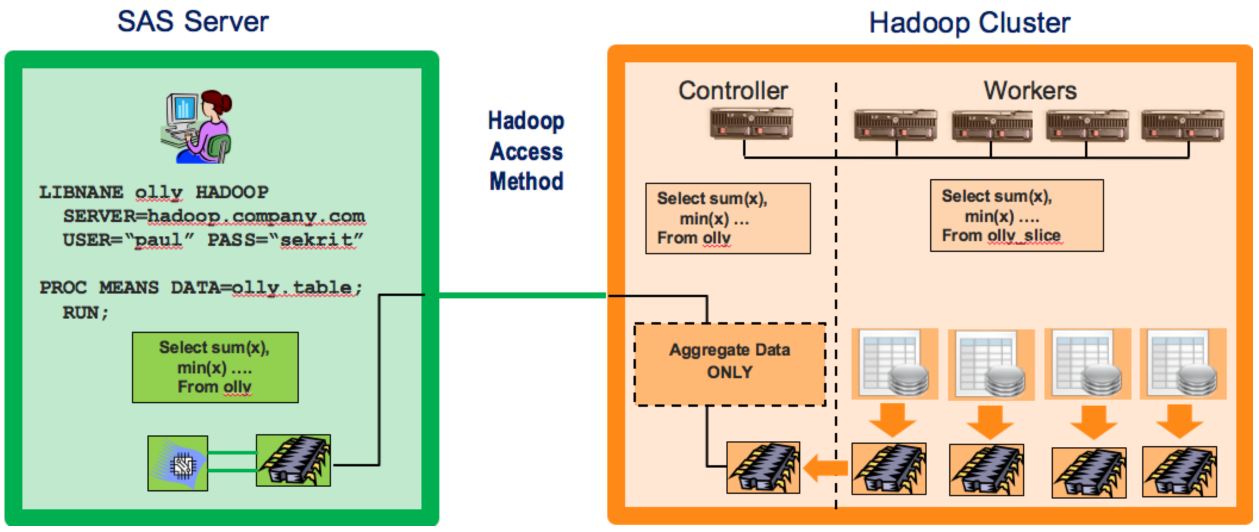
SAS's PROC MEANS procedure calculates simple descriptive statistics. Standard SQL has the SUM(), COUNT() and other functions to calculate the same statistics. What if we could teach PROC MEANS to ask the data source to do the heavy lifting, rather than just access the raw records and perform the calculations itself. We have come to call the former "SQL passthru" – and it is the same concept as the MC at the wedding asking the tables themselves to determine the youngest at each table – transfer a high level specification of the work rather than request all the raw materials be delivered to your doorstep!

SAS ACCESS to Hive implements this concept, and as these two diagrams demonstrate, can have a great impact on the number of rows of data that must be transferred between the Hadoop server and the SAS Session

In the default scenario, PROC MEANS must request all the rows from the table for which you require the descriptive statistics. It doesn't matter that HIVE can do a nice job of partitioning the work and make cluster-local request for the data slice that lives on that cluster – eventually all rows in the table have to be sent over the connection back to SAS:



In the SQL passthru scenario, things are a lot better. The request to PROC MEANS is understood and translated to the equivalent SQL asking for those statistics. The HIVE optimizer can distribute the work optimally across the cluster, and a small set or records is returned to the SAS server for the result. We are able to take advantage of Hadoops' Idea#2 – move the work to the data!



This SQL pushdown can get quite sophisticated. Here is an example from PROC RANK against an Impala data source.

```
options sqlgeneration=dbms;
proc rank data=x.cake out=order descending ties=low;
   var present taste;
   ranks PresentRank TasteRank;
run;
```

You will see the generated SQL in your SAS log:

```
IMPALA_25: Prepared: on connection 2
SELECT `table0`.`name`, `table0`.`present`, `table0`.`taste`,
`table1`.`rankalias0` AS `PresentRank`, `table2`.`rankalias1` AS
`TasteRank` FROM ( SELECT `name` AS `name`, `present` AS `present`, `taste`
AS `taste` FROM `cake` ) AS `table0` LEFT JOIN ( WITH subquery0 AS (SELECT
`present`, `tempcol0` AS `rankalias0` FROM ( SELECT `present`, MIN(
`tempcol1` ) OVER ( PARTITION BY `present` ) AS `tempcol0` FROM( SELECT
`present`, CAST( ROW_NUMBER() OVER ( ORDER BY `present` DESC ) AS DOUBLE )
AS `tempcol1` FROM ( SELECT `name` AS `name`, `present` AS `present`,
`taste` AS `taste` FROM `cake` ) AS `subquery2` WHERE ( ( `present` IS NOT
NULL ) ) ) AS `subquery1` ) AS `subquery0` ) SELECT DISTINCT `present`,
`rankalias0` FROM subquery0 ) AS `table1` ON ( (`table0`.`present` =
`table1`.`present` ) ) LEFT JOIN ( WITH subquery3 AS (SELECT `taste`,
`tempcol2` AS `rankalias1` FROM ( SELECT `taste`, MIN( `tempcol3` ) OVER (
PARTITION BY `taste` ) AS `tempcol2` FROM( SELECT `taste`, CAST(
ROW_NUMBER() OVER ( ORDER BY `taste` DESC ) AS DOUBLE ) AS `tempcol3` FROM
( SELECT `name` AS `name`, `present` AS `present`, `taste` AS `taste` FROM
`cake` ) AS `subquery5` WHERE ( ( `taste` IS NOT NULL ) ) ) AS `subquery4`
) AS `subquery3` ) SELECT DISTINCT `taste`, `rankalias1` FROM subquery3 )
AS `table2` ON ( ( `table0`.`taste` = `table2`.`taste` ) )
```

Thank goodness SAS is taking care of all the nasty details of translating your innocent looking PROC RANK request into the appropriate SQL that takes care of the correct ranking even in the face of ties and missing values.

SPDE – ASK HADOOP TO STORE SAS DATASETS

Hadoop has HDFS – which is great at storing files and keeping them safe. Can I store my SAS datasets there?

There is a flavor of SAS dataset that is well suited to storing in Hadoop – the SPDE format.

```
libname spdat spde '/user/dodeca' hdfshost=default;
```

This allows you to save your SAS datasets in the Hadoop cluster. Your CFO will like this, as storage costs inside the Hadoop cluster are typically an order of magnitude less than what SAN storage costs. This feature is new as of SAS 94m3, and:

- Supports the SAS code accelerator
- Allows for enhanced WHERE pushdown: AND, OR, NOT, parenthesis, range operators and in-lists
- Parallel write support can improve write performance up to 40%
- Optionally uses Apache Curator/Zookeeper as a distributed lock server. No more physical lock files.
- Provides a Hadoop SerDe for direct read access to SPDE data from the Hadoop ecosystem
- Provides a Java tool to populate the Hive metastore with SPDE metadata

So the second level of connectivity between SAS and Hadoop is SAS Access Software and its connection to Hive, Impala and Hawq. It facilitates the transfer of richer metadata regarding tables, as well as encourages the “hadoop way” – move the work to the data, by means of SQL passthru. The second level also includes the SPDE format for Hadoop, which allows you to save your SAS datasets in Hadoop

SAS AND HADOOP – GETTING ENGAGED

Expressing work as SQL and pushing that down to the data source is well and good, but taking it to the next level requires that most all of the SAS Syntax gets executed on the cluster. There are two patterns for this:

1. The SAS® Code Accelerator for Hadoop, which is an embedded process that can run DS2 code
2. The SAS High Performance Procedures (and the LASR server behind SAS® Visual Analytics)

THE SAS CODE ACCELERATOR

You can think of the SAS Code Accelerator as the runtime engine for processing SAS DS2 code. This engine exists on every Data Node in the cluster, and so can process requests with the same degree of parallelism as the cluster itself. The source code for programs run by the Code Accelerator might be models published from SAS Analytical Procedures, routine data management tasks as generated by SAS Data Loader for Hadoop, or they might be specialty tasks for the data quality transformations.

New as of SAS 94m3 is support for the MERGE statement. DS2 generates Hive SQL to emulate the classic DATA Step merge, and your data flows into the DS2 program data vector in the traditional BY Group style – so your programs have access to IN= as well as FIRST. And LAST. Functionality

The SAS Code Accelerator for Hadoop can take its input from several sources:

- SAS SPDE format datasets stored in HDFS
- Hive tables defined to Hadoop
- ORC, Parquet, Avro tables defined via the HCATALOG

The DS2 Language supported by the SAS Code Accelerator for Hadoop supports a rich variety of input processing

- Arbitrary SQL on the SET statement

```
set { select * from t1 inner join t2 on t1.id = t2.id };
```

- Multiple tables on the SET statement are emulated with HIVE UNION ALL syntax
- MERGE statement is supported by generating custom HIVE SQL to handle the grouping as well as FIRST. and LAST. indicator variables

The SAS Code Accelerator for Hadoop runs inside a Hadoop MapReduce job. As of SAS 94m3, it is fully integrated with the YARN resource manager

THE SAS HIGH PERFORMANCE ARCHITECTURE PROCEDURES

As of SAS 94m3, a significant set of SAS procedures have been rewritten to run is a distributed fashion on a cluster.

The 2nd generation of SAS’s distributed computing is embodied in the LASR server behind SAS® Visual Analytics and other SAS Software. This server holds your dataset in the distributed memory of the cluster, and contains a set of actions that can be executed against the datasets. You can well imagine there must be actions that correspond to the tasks in Visual Analytics, like for regression, correlation and forecasting. You can get a feel for the actions more closely from the language of the SAS In Memory Statistics software.

SAS AND HADOOP – GETTING MARRIED

Three levels of commitment described in this paper so far:

1. Exchanging raw files with HDFS;
2. Exchanging table-level metadata and using SQL to push work into the cluster;
3. the scoring accelerator and SAS High Performance Architecture for calculating the results of SAS procedures in a distributed fashion.

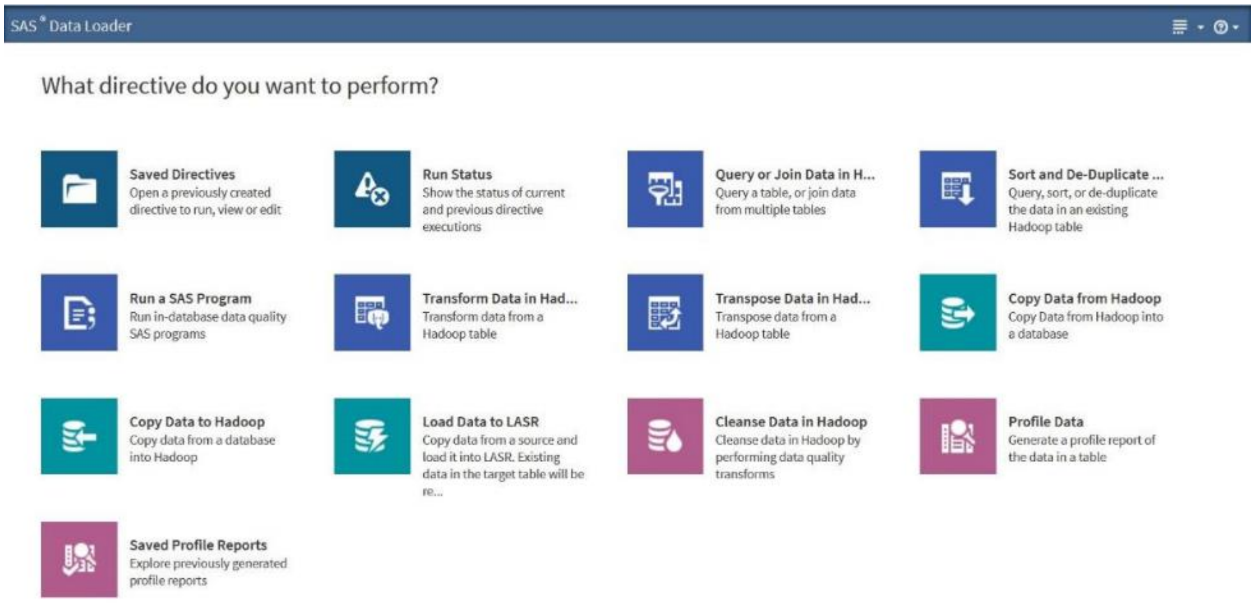
These are significant building blocks that allow SAS to build complete products that are based on and in a Hadoop Cluster. Hadoop is simply one of the requirements – you may install the SAS product on your existing cluster, or you might use a purpose built cluster and the Hadoop distribution included with SAS to support your application.

Examples of SAS Software built on this exciting new architecture are:

1. SAS® Data Loader for Hadoop
2. SAS® Visual Analytics and Visual Statistics
3. SAS® In Memory Statistics
4. SAS® Grid Manager for Hadoop

There are many other papers with detail on these products from SAS. I include a screen shot or code snippet here to give you a flavor of what is possible as we begin to build on top of this new modern analytical platform.

SAS DATALOADER FOR HADOOP



SAS® Data Loader

What directive do you want to perform?

Saved Directives
Open a previously created directive to run, view or edit

Run Status
Show the status of current and previous directive executions

Query or Join Data in H...
Query a table, or join data from multiple tables

Sort and De-Duplicate ...
Query, sort, or de-duplicate the data in an existing Hadoop table

Run a SAS Program
Run in-database data quality SAS programs

Transform Data in Had...
Transform data from a Hadoop table

Transpose Data in Had...
Transpose data from a Hadoop table

Copy Data from Hadoop
Copy Data from Hadoop into a database

Copy Data to Hadoop
Copy data from a database into Hadoop

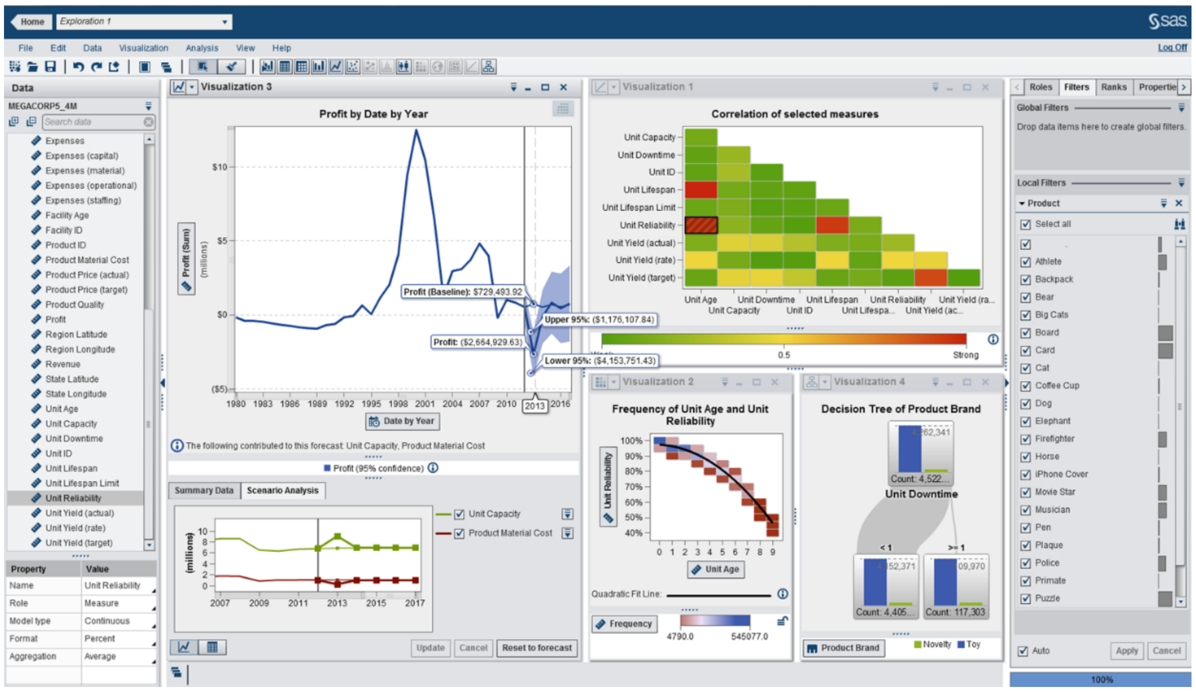
Load Data to LASR
Copy data from a source and load it into LASR. Existing data in the target table will be re...

Cleanse Data in Hadoop
Cleanse data in Hadoop by performing data quality transforms

Profile Data
Generate a profile report of the data in a table

Saved Profile Reports
Explore previously generated profile reports

SAS VISUAL ANALYTICS

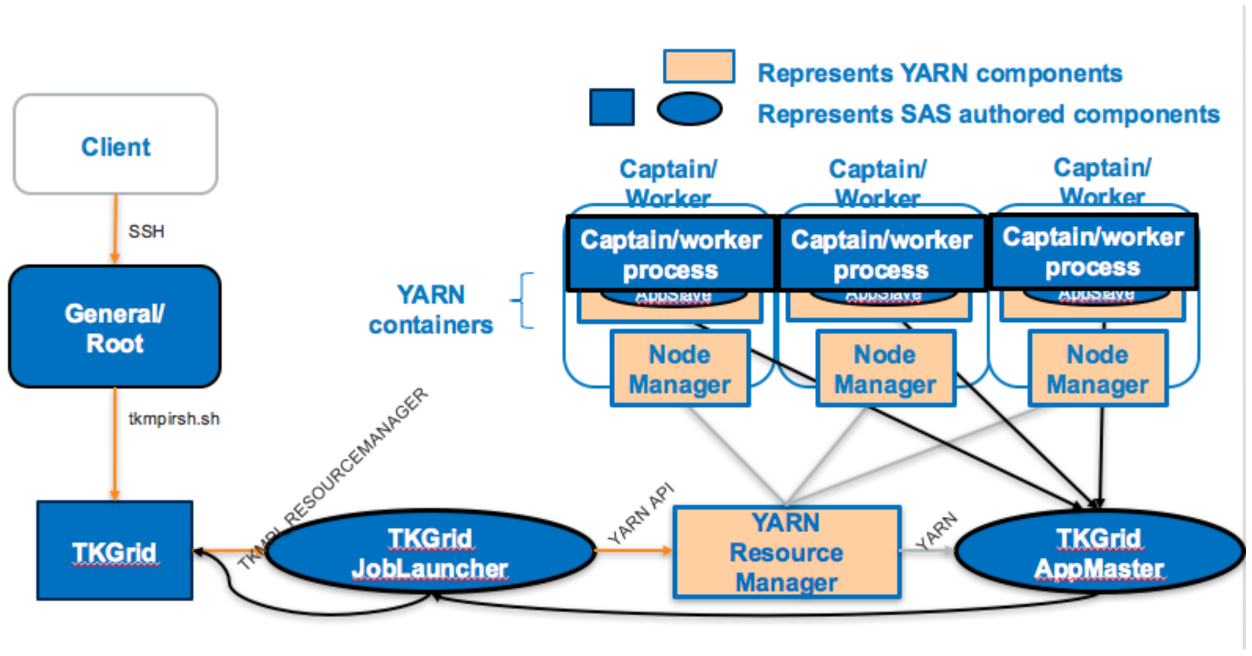


Exploration 1
File Edit Data Visualization Analysis View Help
Data
MEGACORP5_4M
Visualization 3
Profit by Date by Year
Profit (Baseline): $729,493.92
Profit: ($2,664,929.63)
Upper 95%: ($1,176,107.84)
Lower 95%: ($4,153,751.43)
Date by Year
The following contributed to this forecast: Unit Capacity, Product Material Cost
Summary Data
Scenario Analysis
Unit Capacity
Product Material Cost
Update Cancel Reset to forecast
Visualization 1
Correlation of selected measures
Visualization 2
Frequency of Unit Age and Unit Reliability
Quadratic Fit Line
Visualization 4
Decision Tree of Product Brand
Unit Downtime
Roles Filters Ranks Properties
Global Filters
Local Filters
Product
Apply Cancel

SAS IN MEMORY STATISTICS

```
proc imstat;
    /*-- list tables in the server -----------------*/
    tableinfo / save=tmpcontent;
    store tmpcontent(2,3) = numObs;
run;
    /*-- print the last ten records from carinfo ---*/
    table lasr.carinfo;
    fetch / from=%eval(&numObs.-9) to=&numObs;
run;
    /*-- working on the fact table (cardata) -------*/
    /*-- data exploration using different actions --*/
    table lasr.cardata;
         distributioninfo;
         distinct _all_;
         boxplot mmrcurrentauctionaverageprice
                 mmrcurrentauctioncleanprice
                 mmrcurrentretailaverageprice
                 mmrcurrentretailcleanprice /
         groupby=auction;
         frequency isbadbuy;
         crosstab isbadbuy*vehyear / groupby=auction;
run;
    table lasr.cardata(tempnames=(avgOdo));
         summary  avgOdo
           / tempnames=avgOdo
             tempexpress="avgOdo = vehodo / (year(purchdate)-vehyear);";
run;

proc recommend port=&lasrport;
  add / item=movieid
  user=userid
  rating=rating;
  addtable mylasr.ml100k / type=rating vars=(movieid userid rating);

  method svd / factors=10 maxiter=20 maxfeval=100 tech=lbfgs seed=1234
               function=L2 lamda=0.1 label="svdlbfgs" details;

  method knn / similarity=pc positive k=20 label="knn1";

  method ensemble / details label="em1" maxiter=10 seed=4321;
run;
```

SAS GRID MANAGER FOR HADOOP



RELEASE NUMBER SOUP

The Hadoop eco-system releases new versions of the many sub-projects that make up a Hadoop distribution at a rapid and sometimes alarming pace. This can often be a challenge for a Software Vendor like SAS. We keep track of the supported versions of various distributions at this web page:

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

CONCLUSION

Hadoop has most definitely changed the analytics landscape. Building models using a large number of computers using distributed computing concepts has become commonplace, as has scoring those models in an “as close to real time” fashion.

SAS Software has evolved to keep up with these trends.

ACKNOWLEDGMENTS

Many people across SAS Institute contribute towards the features that support interacting with Hadoop.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact the author at:

Paul Kent
SAS Institute, Inc.

paul.kent@sas.com