SAS® FORUM
UNITED KINGDOM 2015

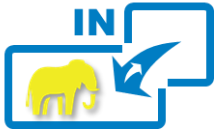
Hadoop – review of SAS' Strategy & technology

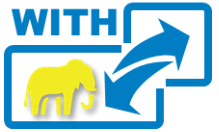
Doug Green

Review of integration points

SAS & Hadoop intersect in many ways:

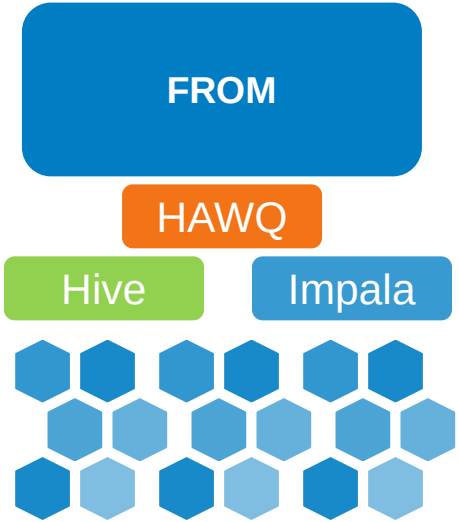
- SAS can treat Hadoop just as any other data source, pulling data **FROM** Hadoop, when it is most convenient;
- SAS can work directly **IN** Hadoop, leveraging the distributed processing capabilities of MapReduce.
- SAS can work directly **WITHIN** Hadoop, lifting data on HDFS into a SAS advanced analytics in-memory environment;



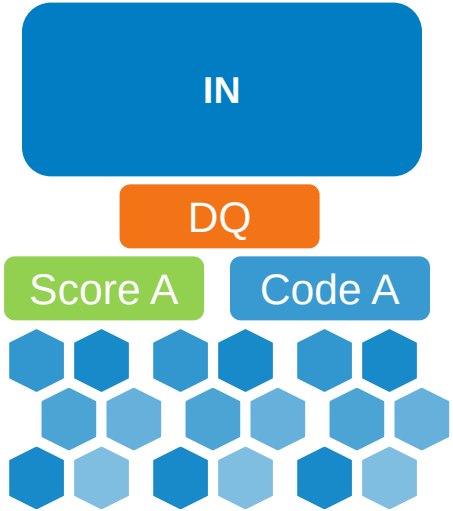


Review of SAS' technology direction

SAS/Access to Hadoop - Extract data from Hadoop into SAS



Embedded Process - Push *some* SAS processing to Hadoop with Map Reduce



In-Memory Analytics - Use Hadoop for Storage persistence, workload mgmt. and commodity computing.

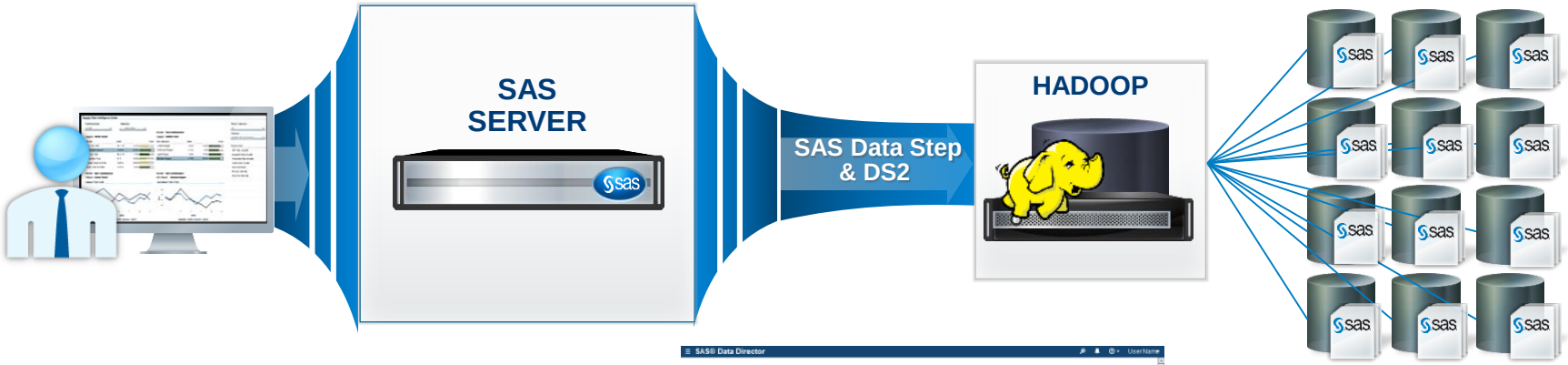
Access engines overview

SAS/Access to Hadoop, Hawq or Impala - Push *some* of SAS' processing to Hadoop

SAS/Access to Hadoop
SAS/Access to Cloudera Impala
SAS/Access to HAWQ (Pivotal HD, Hortonworks)

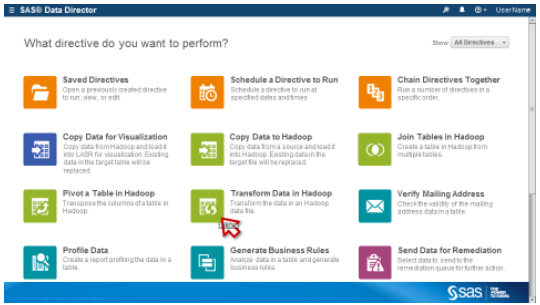
SAS Accelerators for Hadoop overview

SAS/Embedded Process - Push SAS processing to Hadoop with Map Reduce



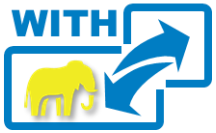
SAS/Scoring Accelerator for Hadoop
SAS/Data Loader for Hadoop

- SAS/Code Accelerator for Hadoop
- SAS/Data Quality Accelerator for Hadoop



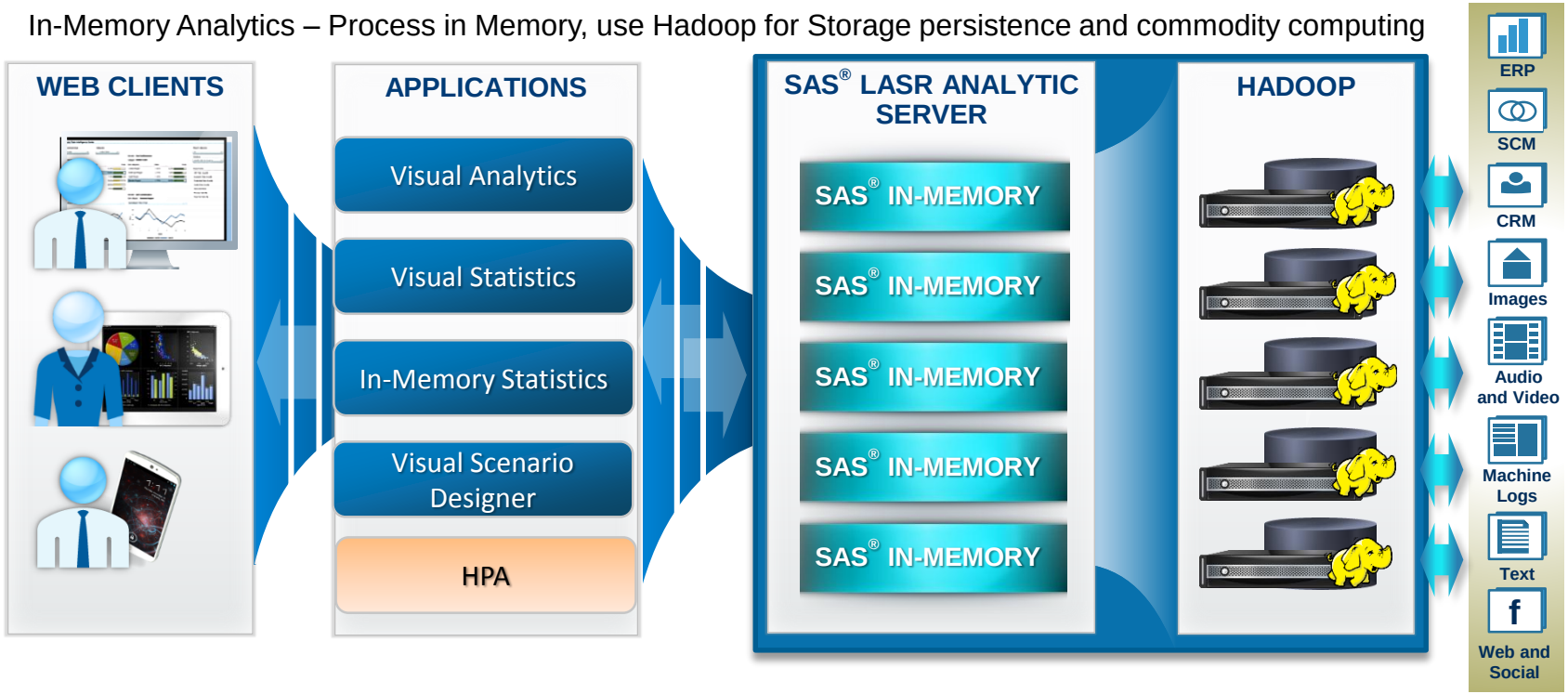

```
proc ds2 ;
/* thread ~ eqiv to a mapper */
   thread map_program;
   method run(); set dbmslib.intab;
   /* program statements */
   end; endthread; run;
   /* program wrapper */
   data hdf.data_reduced;
dcl thread map_program map_pgm;
method run();
set from map_pgm threads=N;
/* reduce steps */ end; enddata;
run; quit;
```

In-memory analytics overview



SAS ANALYTIC HADOOP ENVIRONMENT

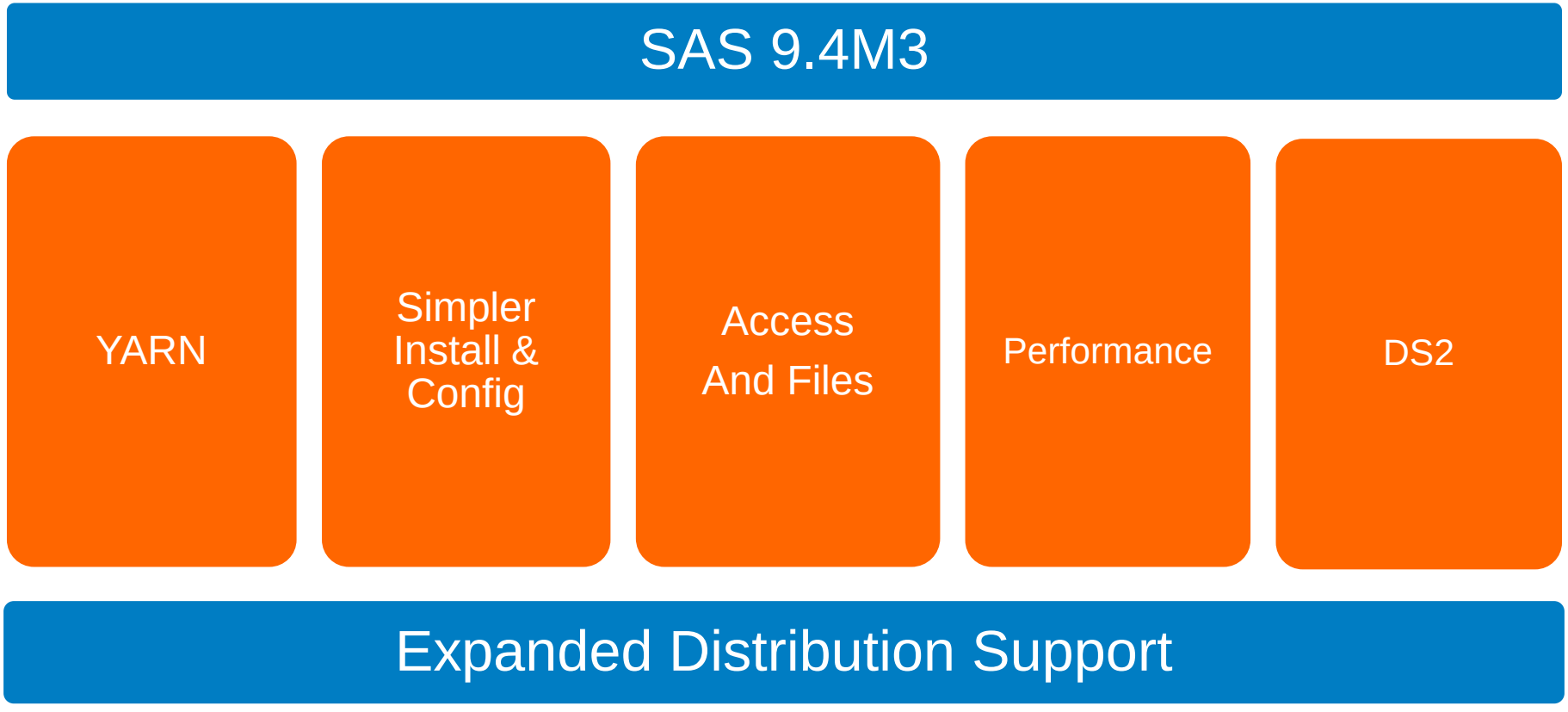
In-Memory Analytics – Process in Memory, use Hadoop for Storage persistence and commodity computing



SAS® FORUM
UNITED KINGDOM 2015

What's coming for Hadoop in July with 9.4 M3

Major Hadoop themes for SAS 9.4 M3



Tighter YARN integration

- Continue the momentum of 9.4M2 where LASR & HPA first became YARN aware.
- SAS EP now runs in same process as the MapReduce JVM – more tightly coupled resource
  - Fully integrated with YARN Resource Manager
  - SAS EP can now write to MR job logs through put statements
  - Performance improvements over M2

Easier deployment

Simpler Install & Config

- SAS Deployment wizard can now gather required jars
- Integration of SAS EP with Hadoop distributor administration tools to simplify SAS install/configuration
  - integration with Cloudera Manager and Ambari
  - Packages and parcels
  - No root required – SUDO only
  - Does *not* require two way SSH keys setup

Access engine improvements

- SAS/ACCESS to Hadoop
  - Support for BINARY and DECIMAL data types
  - Implicit Pass-through improvements
    - READ_METHOD=HDFS honored
  - CTRL-C Query Interrupt
  - Improved error messaging
- SAS/Access to HAWQ including Proc pushdown support
- SAS/Access to Impala adds Proc pushdown support

File type support

Access and Files

- Increased ability to read and write Hadoop file types using SAS EP
  - Full EP support for Parquet, Avro, JSON, ORC and compressed Sequence files.
- SerDe's to make reading SAS proprietary file types easy for Hadoop Community e.g. SPDE, SASHDAT
- PROC SQOOP - GA

Expanded distribution support

Distributions

- Expand SAS' Distribution Support (SAS/Access, EP,HPA,LASR)
  - Near Parity between CDH / HWX and MapR, Pivotal and IBM
  - New HAWQ access Engine

Performance

- Expand Hadoop Pushdown Processing
  - Data Step / DS2 – Merge, Proc Transpose
- SPDE on HDFS
  - Improved WHERE pushdown: AND, OR, NOT, parenthesis, range operators and in-lists all supported
  - Parallel write support can improve write performance up to 40%

New products with SAS 9.4 M3

SAS Grid

- SAS Grid Manager for YARN
  - Hadoop support is an alternative, not a replacement for LSF

SPDS

- SPDS supports Hadoop (5.2)
  - Ability to read, write and update SPD Server tables stored in Hadoop
  - Kerberos only

SAS® FORUM
UNITED KINGDOM 2015

An Introduction to DS2 on Hadoop

Doug Green

sas THE POWER TO KNOW.

What is DS2?

- DATA step like distributed processing for Hadoop and other MPP platforms (think do loops, arrays, statements, by-group processing, functions etc.)
- Appropriate for advanced data manipulation and data modelling applications especially those that are difficult or impossible to achieve through SQL (e.g. transposing data)
- Object orientated programming environment
- Runs in the MapReduce/YARN framework on Hadoop
- Portable across platforms

The SAS Embedded Process

A portable, lightweight execution container for SAS code that makes SAS portable and deployable on a variety of platforms

```
OPTIONS DS2ACCEL=ANY DSACCEL=ANY;
proc ds2 ;

/*---
**MAP
*/
thread map_program / overwrite=yes;
method run();
set Hadoop.&source;
/* program statements */
end;
endthread;

/*------
**REDUCE
*/
data hadoop.&target overwrite=yes);
dcl thread map_program MapReduce;
method run();
set from MapReduce;
end;
enddata;
run;
quit;
```

1. Data Lifting
2. Data Preparation
3. Data Quality
4. Scoring

SAS® FORUM
UNITED KINGDOM 2015

A simple example for Hadoop

DS2 syntax framework for Hadoop

```
1
2  /*HIVE libname */
3  libname hadoop hadoop SUBPROTOCOL=hive2 READ_METHOD=HDFS schema=sukdmg user=sukdmg pwd="{SAS002}E043FE4757B4CE074DC2458F2E9204C53282784D2A0DA252
4  server="XXXXXXXXXXXXX" port=10001 ;
5
6
7  %let source=HADOOP_SOURCE_TABLE:
8  %let target=HADOOP_TARGET_TABLE;
9
10 OPTIONS DS2ACCEL=ANY DSACCEL=ANY;
11 proc ds2 ;
12
13 /*---
14 **MAP PHASE
15 */
16 thread map_program / overwrite=yes;
17 method run();
18 set Hadoop.&source;
19 /* DS2 program statements */
20 end;
21 endthread;
22
23 /*------
24 **REDUCE PHASE (If by Statement used)
25 */
26 data hadoop.&target overwrite=yes);
27 dcl thread map_program MapReduce;
28 method run();
29 set from MapReduce;
30 end;
31 enddata;
32 run;
33 quit;
```

1. Hadoop libname
2. SAS Options
3. Create thread program
4. DS2 logic
5. Call thread program

An important option!

- For any DS2 thread program to run in Hadoop the following SAS option must be set:

```
21
22 options DS2ACCEL=ANY;
23
```

Original data step program

```
data test;
  input i j x;
datalines;
1 1 123
1 1 3245
1 2 23
1 2 543
1 2 87
1 3 90
2 1 88
2 1 86
;


/* When the first observation in each BY-Group is read, the variables JSUB and  */
/* FREQ are initialized to zero and with each subsequent observation in the     */
/* BY-Group, FREQ is incremented by one and JSUB is incremented by the value of */
/* X. When the last observation in the BY-Group is read, AVER is created by     */
/* dividing JSUB by FREQ to determine the average value for the group.          */
```

```
data jsubtot (keep=i j freq aver);
  set test;
  by i j;
  retain jsub freq;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
run;
```

```
proc print;
run;
```

Obs	i	j	freq	aver
1	1	1	2	1684.00
2	1	2	3	217.67
3	1	3	1	90.00
4	2	1	2	87.00

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

Obs	freq	aver	I	j
1	2	87.00	2	1
2	2	1684.00	1	1
3	3	217.67	1	2
4	1	90.00	1	3

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

DS2 is a SAS procedure and is therefore invoked through SAS procedure syntax.

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

To run in-database, a thread program must be used. The SAS Code Accelerator enables you to publish a DS2 thread program and execute that thread program in parallel inside Hadoop.

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

Unlike Base/SAS, DS2 enables you to explicitly declare variables using the DECLARE statement. Here it is declared outside of a method so its scope is GLOBAL.

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

DS2 has new data types, more akin to an RDBMS, and should be explicitly declared. E.g. VARCHAR, DOUBLE, INT, BIGINT etc.

DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

DROP/KEEP/RETAIN/RENAME are only valid in global scope. i.e. outside of a method programming block.

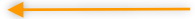
DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

Method run() is a system method – will execute in an implicit loop for every row of the input data. Other system methods are init() & term()

DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```



This block of code is identical to the original data step program.

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

A BY statement is required to generate Hadoop REDUCE tasks. Without a BY statement, only MAP tasks are generated.

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

End statement to close the run() method.

DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

Endthread statement to close the thread program.

DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

Now we reference the output dataset to be created on Hadoop

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

Explicitly declare the thread program and specify a name that identifies an instance of the thread.

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

Use method run() to allow the program to read from the thread program

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

Read the thread program by referencing the thread identifier

DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

End statement to close the run() method.

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

The enddata statement marks the end of a data statement

DS2 equivalent for Hadoop

```
proc ds2;
     thread compute / overwrite=yes;
          declare double jsub freq aver;
          retain jsub freq;
          keep i j freq aver;
          method run();
   set hdp.test;
   by i j;
   if first.j then do;
     jsub=0;
     freq=0;
   end;
   jsub + x;
   freq + 1;
   if last.j then do;
     aver=jsub/freq;
     output;
   end;
   end;
endthread;
data hdp.jsubtot (overwrite=yes);
     declare thread compute t;
     method run();
     set from t;
     end;
     enddata;
run;
quit;
```

The RUN statement submits the DS2 statements

DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

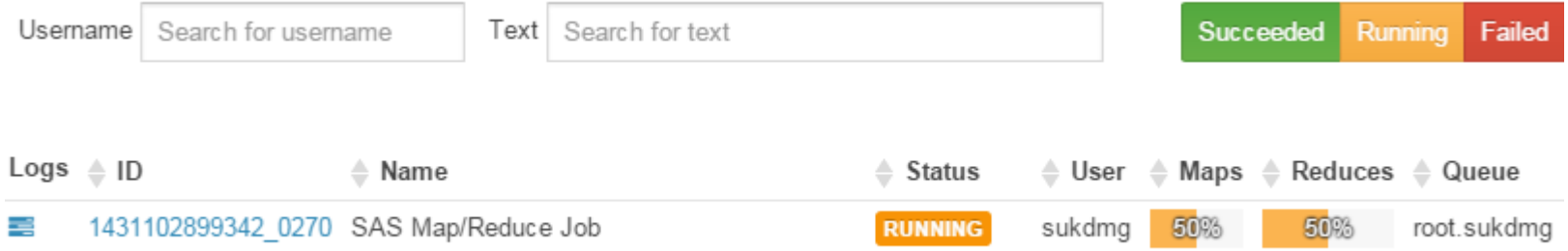
As DS2 is a SAS procedure we must explicitly quit it

The SAS Log

```
83
84          proc ds2;
NOTE: Connection string:
NOTE: DRIVER=DS2;CONOPTS= (DRIVER=FEDSQL;CONOPTS= ( (DRIVER=base;CATALOG=WORK;SCHEMA=
      (NAME=WORK;PRIMARYPATH={/tmp/SAS_work468600004F68_ukva1-01.suk.sas.com/SAS_work415E00004F68_ukva1-01.suk.sas.com}));
      (DRIVER=HIVE;SERVER=gbrhadoop1-01.suk.sas.com;UID=sukdmg;PWD=*;PORT=10001;SUBPROTOCOL=hive2;HD_CONFIG=/tmp/SAS_work468600004F6
      8_ukva1-01.suk.sas.com/#LN02581;SCHEMA=sukdmg;CATALOG=HDP); (DRIVER=base;CATALOG=WEBWORK;SCHEMA=
      (NAME=WEBWORK;PRIMARYPATH={/home/sukdmg/.WebWork})); (DRIVER=base;CATALOG=SASDATA;SCHEMA=
      (NAME=SASDATA;PRIMARYPATH={/data/SAS/config/Lev1/SASApp/Data})); (DRIVER=base;CATALOG=STPSAMP;SCHEMA=
      (NAME=STPSAMP;PRIMARYPATH={/data/SAS/software/SASFoundation/9.4/samples/inttech})); (DRIVER=base;CATALOG=VALIB;SCHEMA=
      (NAME=VALIB;PRIMARYPATH={/data/SAS/config/Lev1/SASApp/Data/valib})); (DRIVER=base;CATALOG=MAPS;SCHEMA=
      (NAME=MAPS;PRIMARYPATH={/data/SAS/software/SASFoundation/9.4/maps})); (DRIVER=base;CATALOG=MAPSSAS;SCHEMA=
      (NAME=MAPSSAS;PRIMARYPATH={/data/SAS/software/SASFoundation/9.4/maps})); (DRIVER=base;CATALOG=MAPSGFK;SCHEMA=
      (NAME=MAPSGFK;PRIMARYPATH={/data/SAS/software/SASFoundation/9.4/mapsgfk})); (DRIVER=base;CATALOG=SASUSER;SCHEMA=
      (NAME=SASUSER;PRIMARYPATH={/home/sukdmg/sasuser.v94})))))
85          thread compute / overwrite=yes;
86          declare double jsub freq aver;
87           retain jsub freq;
88          keep i j freq aver;
89          method run();
90            set hdp.test;
91            by i j;
92            if first.j then do;
93              jsub=0;
94              freq=0;
95            end;
96            jsub + x;
97            freq + 1;
98            if last.j then do;
99              aver=jsub/freq;
100             output;
101           end;
102           end;
103         endthread;
104         data hdp.jsubtot (overwrite=yes);
105         declare thread compute t;
106         method run();
107         set from t;
108         end;
109         enddata;
110         run;
NOTE: Created thread compute in data set work.compute.
NOTE: Running THREAD program in-database
NOTE: Running DATA program in-database
NOTE: Execution succeeded. No rows affected.
111         quit;

NOTE: PROCEDURE DS2 used (Total process time):
```

Obs	freq	aver	i	j
1	2	87.00	2	1
2	2	1684.00	1	1
3	3	217.67	1	2
4	1	90.00	1	3

What's happening on the Hadoop cluster?



Running data step in Hadoop with 9.4 M2

- Only one input and output data set and both must be on Hadoop
- Only a subset of the full DATA step syntax is currently available for parallel execution.
  - Data step logic is converted to DS2 under the covers
- Only functions and formats that are supported by the DS2 language compile successfully.
  - E.g. No LAG or DIF functions
- SAS Statements not currently supported:

- BY (or FIRST. and LAST. variables)
- CONTINUE
- DISPLAY
- FILE
- INFILE
- INPUT
- LEAVE
- MERGE
- MODIFY
- OUTPUT
- PUT
- REMOVE
- RENAME
- REPLACE
- RETAIN
- UPDATE
- WHERE
- WINDOW

An important option!

- For any Data Step program to run in Hadoop the following SAS option must be set:

```
15
16 options DSACCEL=ANY;
17
```

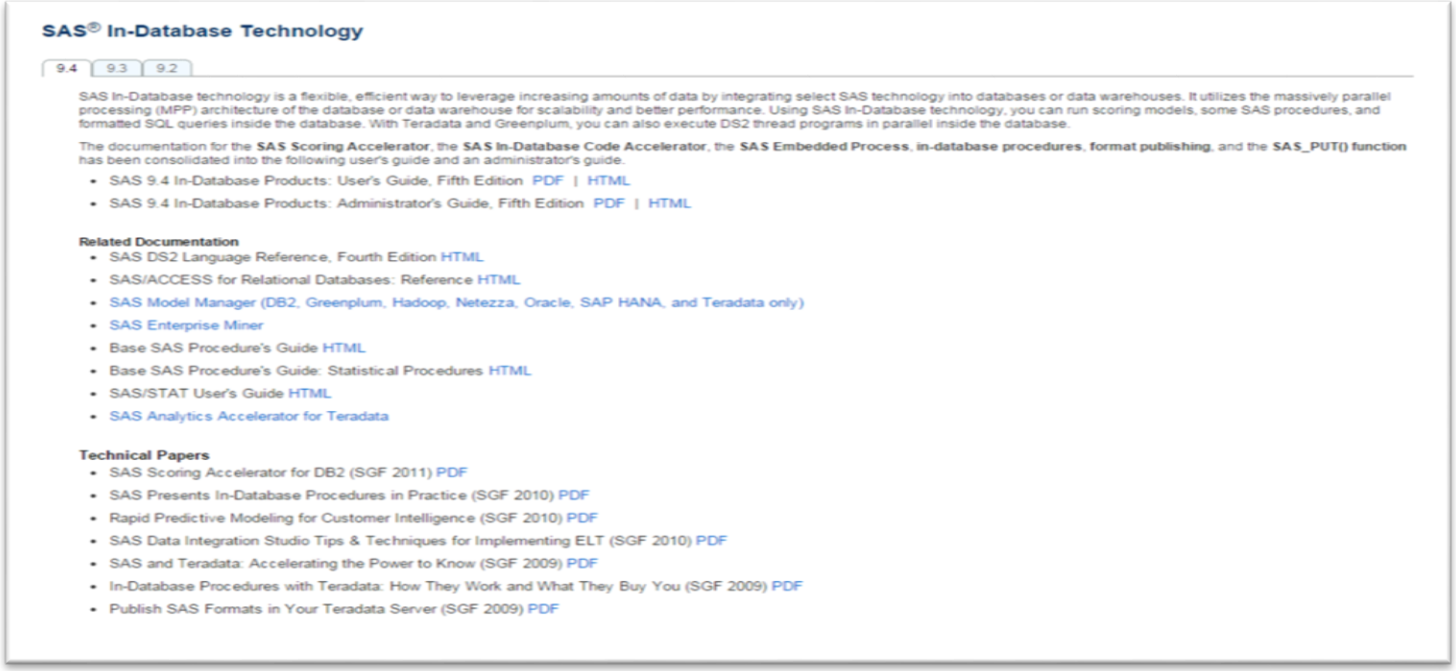
Data Step Example log

```
68          /*Turn on SAS DS1 processing via the SAS EP*/
69          options msglevel=i;
70          options DSACCEL=ANY;
71
72
73          proc delete data=hdfsdemo.scored_big;
74          run;

NOTE: Deleting HDFSDEMO.SCORED_BIG (memtype=DATA).
NOTE: PROCEDURE DELETE used (Total process time):
      real time           0.58 seconds
      cpu time            0.11 seconds


75
76
77          data hdfsdemo.scored_big;
78          set hdfsdemo.intra;
79          /* Execute the score code. */
80          if sum > 1000 then score=1;
81          run;

NOTE: Attempting to run DATA Step in Hadoop.
NOTE: Data Step code for the data set "HDFSDEMO.SCORED_BIG" was executed in the Hadoop EP environment.


          Hadoop Job (HDP_JOB_ID), job_1431102899342_0220, SAS Map/Reduce Job,

http://gbrhadoop1-01.suk.sas.com:8088/proxy/application_1431102899342_0220/
          Hadoop VersionUser
      real time           1:27.45
          2.5.0-cdh5.3.1demo
      cpu time            0.32 seconds


          Started AtFinished At

          May 22, 2015 10:42:23 AMMay 22, 2015 10:43:49 AM
```

What's coming for DS2 for Hadoop with 9.4 M3

- DS2 SET / MERGE
  - Multi-table SET: *set a b c;*
  - SQL SET: *set {select * from A inner join B on A.id = B.id};*
  - MERGE: *merge A B C; by X;*
  - Support for IN=, FIRST. and LAST.

Next steps with DS2



SAS® In-Database Technology

9.4 | 9.3 | 9.2

SAS In-Database technology is a flexible, efficient way to leverage increasing amounts of data by integrating select SAS technology into databases or data warehouses. It utilizes the massively parallel processing (MPP) architecture of the database or data warehouse for scalability and better performance. Using SAS In-Database technology, you can run scoring models, some SAS procedures, and formatted SQL queries inside the database. With Teradata and Greenplum, you can also execute DS2 thread programs in parallel inside the database.

The documentation for the **SAS Scoring Accelerator**, the **SAS In-Database Code Accelerator**, the **SAS Embedded Process**, **in-database procedures**, **format publishing**, and the **SAS_PUT() function** has been consolidated into the following user's guide and an administrator's guide.

- SAS 9.4 In-Database Products: User's Guide, Fifth Edition PDF | HTML
- SAS 9.4 In-Database Products: Administrator's Guide, Fifth Edition PDF | HTML

Related Documentation

- SAS DS2 Language Reference, Fourth Edition HTML
- SAS/ACCESS for Relational Databases: Reference HTML
- SAS Model Manager (DB2, Greenplum, Hadoop, Netezza, Oracle, SAP HANA, and Teradata only)
- SAS Enterprise Miner
- Base SAS Procedure's Guide HTML
- Base SAS Procedure's Guide: Statistical Procedures HTML
- SAS/STAT User's Guide HTML
- SAS Analytics Accelerator for Teradata

Technical Papers

- SAS Scoring Accelerator for DB2 (SGF 2011) PDF
- SAS Presents In-Database Procedures in Practice (SGF 2010) PDF
- Rapid Predictive Modeling for Customer Intelligence (SGF 2010) PDF
- SAS Data Integration Studio Tips & Techniques for Implementing ELT (SGF 2010) PDF
- SAS and Teradata: Accelerating the Power to Know (SGF 2009) PDF
- In-Database Procedures with Teradata: How They Work and What They Buy You (SGF 2009) PDF
- Publish SAS Formats in Your Teradata Server (SGF 2009) PDF

- http://support.sas.com/documentation/onlinedoc/indbtech/

SAS® FORUM
UNITED KINGDOM 2015

Thank You

www.SAS.com