WORKING WITH SAS & HADOOP

DOUG GREEN

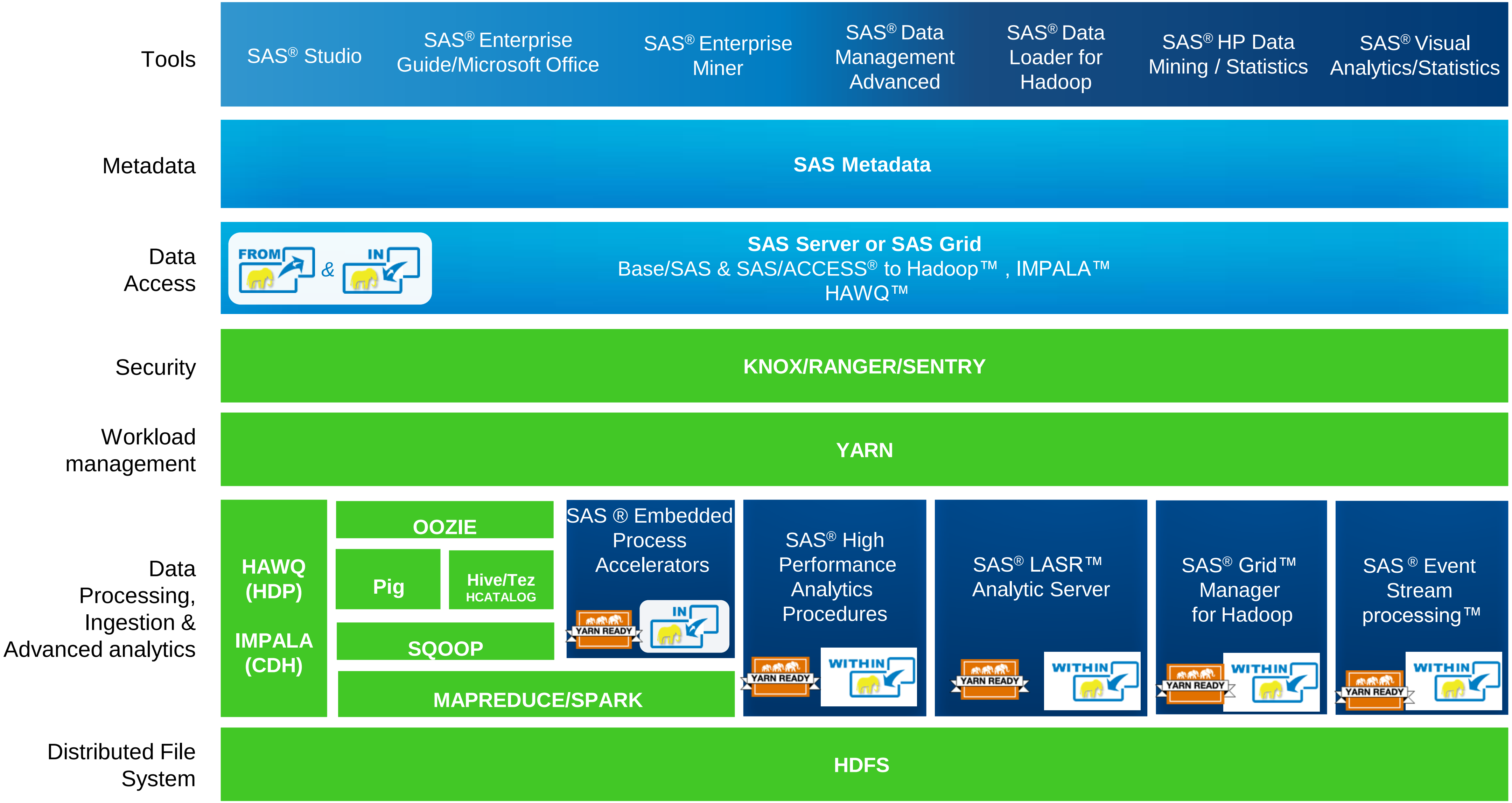
AGENDA

- Review of the *“FROM, IN & WITHIN”* Hadoop integration patterns
- Deployment patterns for SAS HPA/LASR with Hadoop
- SAS/Access and SPDE on HDFS
- DS2 Basics for Hadoop

How does SAS leverage Hadoop?

FROM: *Moving the data out of Hadoop*	Pulling data back to a SAS environment for processing
IN: *Moving the SAS workload to the data*	Run SAS logic in the cluster– process big data with the MapReduce frameworks
WITHIN: *Moving the SAS application to the data*	SAS advanced analytics running natively inside Hadoop under the YARN resource management framework

SAS WITHIN THE HADOOP ECOSYSTEM



Tools: SAS® Studio; SAS® Enterprise Guide/Microsoft Office; SAS® Enterprise Miner; SAS® Data Management Advanced; SAS® Data Loader for Hadoop; SAS® HP Data Mining / Statistics; SAS® Visual Analytics/Statistics

Metadata: SAS Metadata

Data Access: SAS Server or SAS Grid
Base/SAS & SAS/ACCESS® to Hadoop™ , IMPALA™
HAWQ™

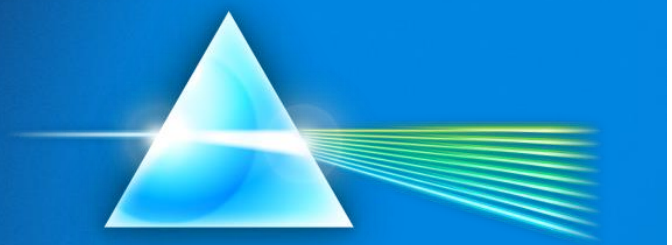
Security: KNOX/RANGER/SENTRY

Workload management: YARN

Data Processing, Ingestion & Advanced analytics: HAWQ (HDP)
IMPALA (CDH); OOZIE; Pig; Hive/Tez HCATALOG; SQOOP; MAPREDUCE/SPARK; SAS ® Embedded Process Accelerators; SAS® High Performance Analytics Procedures; SAS® LASR™ Analytic Server; SAS® Grid™ Manager for Hadoop; SAS ® Event Stream processing™

Distributed File System: HDFS

THE SAS LASR® ANALYTIC SERVER



“It is an in-memory engine specifically engineered for the demands of interactive and iterative analytics”

- In-memory = Fast, sub-second responses
- Multi-User = Hundreds of concurrent users
- Stateless = Don’t pre-compute things
- Interactive = Instantly visualise analytical output
- **Deployment = MPP on HDFS (distributed)** or SMP (single machine)

SAS HIGH PERFORMANCE ANALYTICS (HPA)

```
proc logistic data=HDP.mydata;
   class A B C;
   model y(event='1') = A B B*C;
run;
```

```
proc hplogistic data=HDP.mydata;
   class A B C;
   model y(event='1') = A B B*C;
run;
```

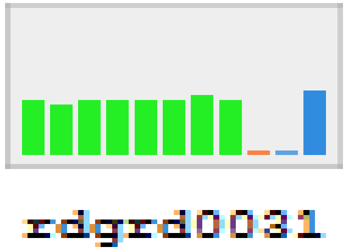
Single / Multi-threaded

Not aware of distributed computing environment

Computes locally / where called

Fetches Data as required

Memory still a constraint



Massively Parallel (MPP)

Two degress of Parrelalism

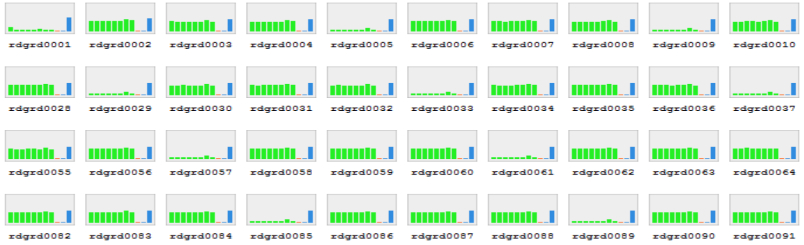
Uses distributed computing environment

Computes in massively distributed mode

Work is co-located with data

In-Memory Analytics

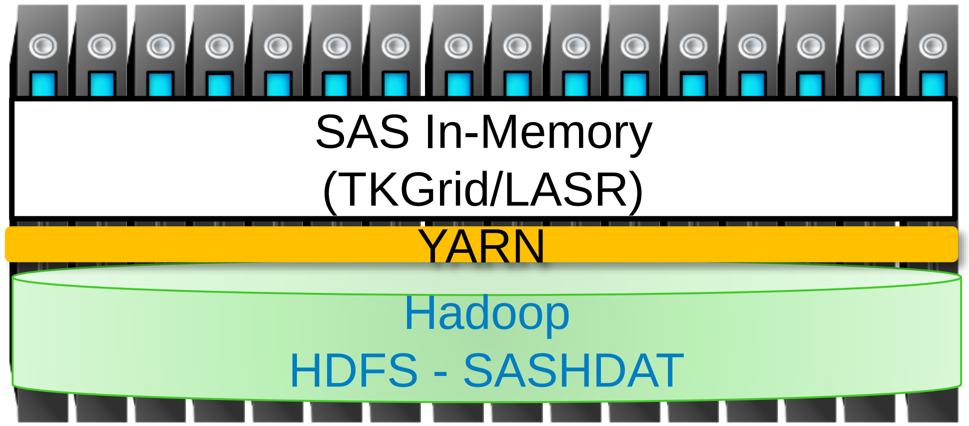
40 nodes x 96GB almost 4TB of memory



LASR VS HPA

	LASR	HPA
Memory Model	Public (Data persisted in-memory and shared)	Private (each execution of the proc creates own copy of the data in-memory. Data is not persisted)
Concurrent users	High	Low
Key SAS Products	• SAS Visual Analytics/Statistics • SAS In-memory statistics	• SAS High Performance Data Mining (via Enterprise Miner) • SAS High Performance statistics (via EG or SAS Studio)

ARCHITECTURE | THREE DEPLOYMENT OPTIONS FOR HPA/LASR



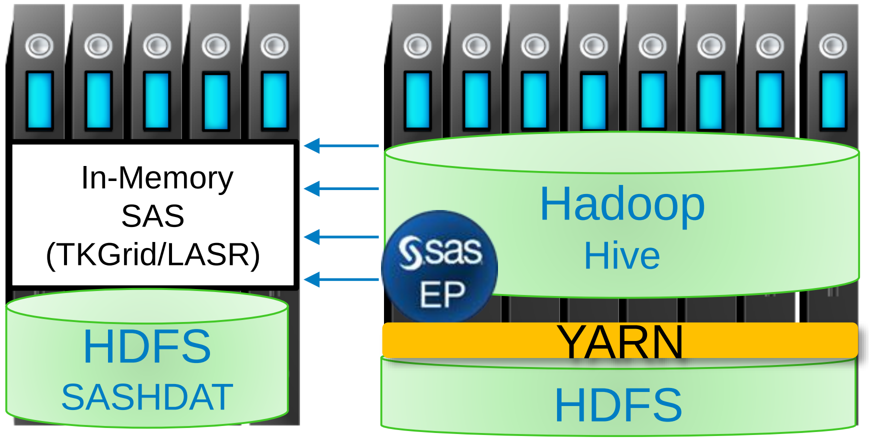
SAS In-Memory
(TKGrid/LASR)

YARN

Hadoop
HDFS - SASHDAT

Symmetric

SAS TKGrid on name and all data nodes



In-Memory
SAS
(TKGrid/LASR)

HDFS
SASHDAT

Hadoop
Hive

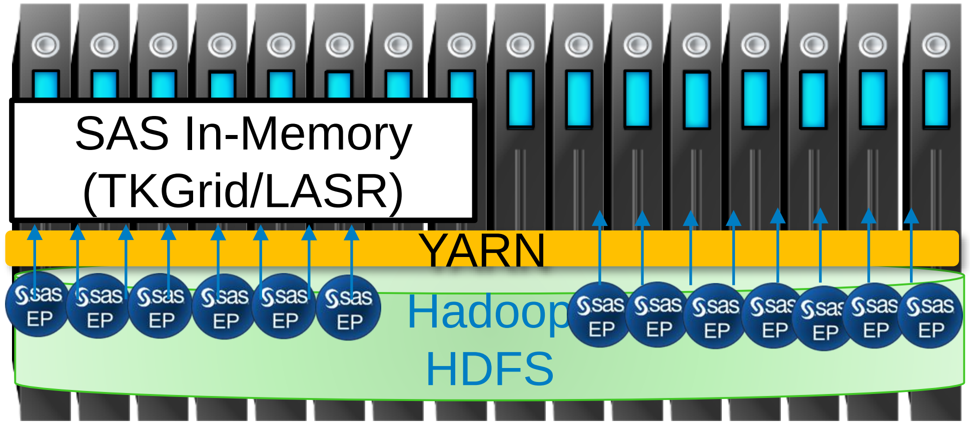
EP

YARN

HDFS

Asymmetric

(separate SAS cluster)



SAS In-Memory
(TKGrid/LASR)

YARN

EP

Hadoop
HDFS

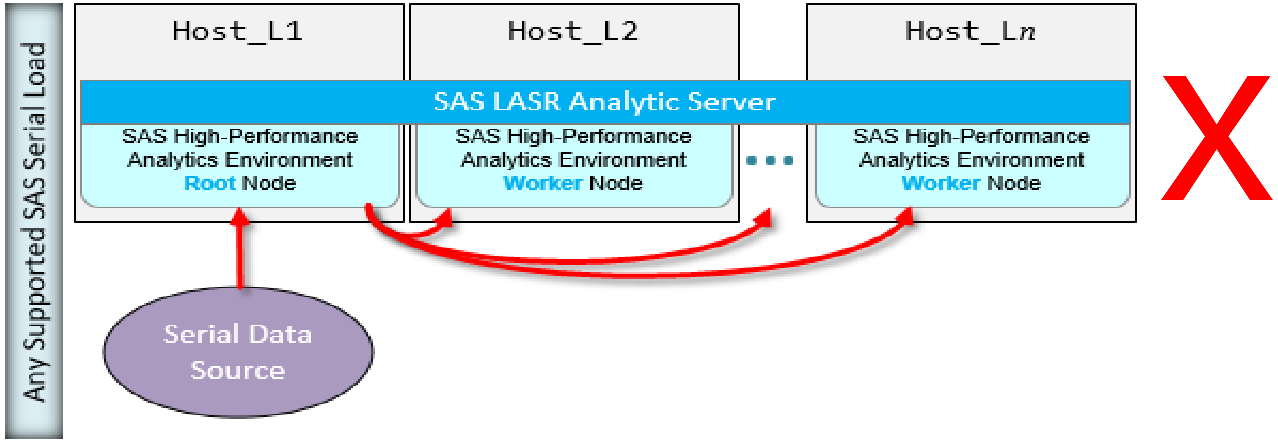
Asymmetric

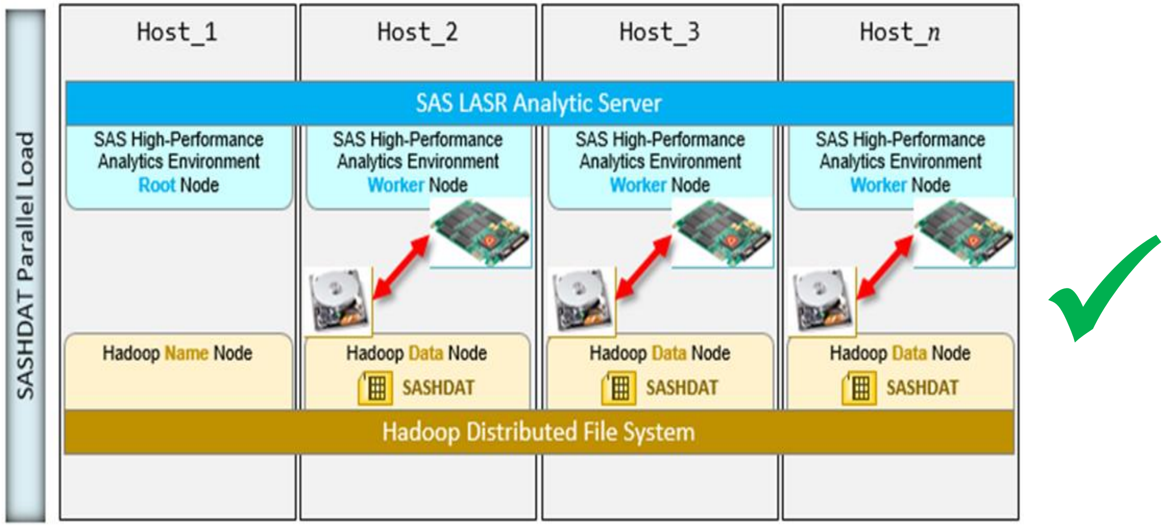
(collocated subset)

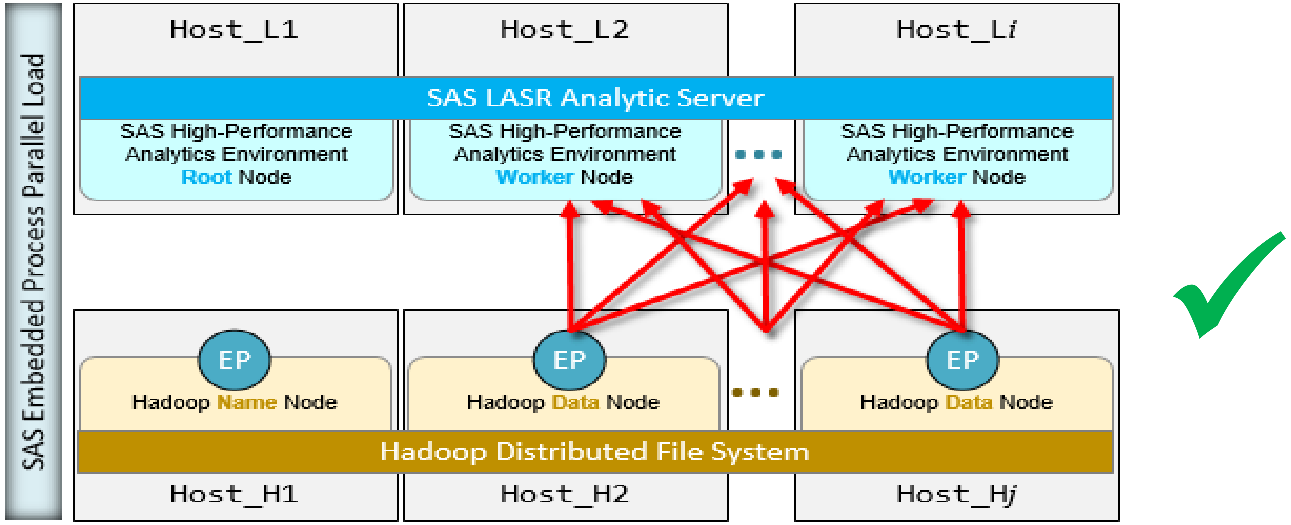
SAS TKGrid on subset of data nodes – YARN manages resources

LOADING DATA INTO HPA/LASR

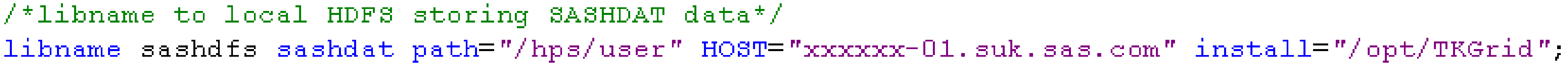
1. Load data from SASHDAT if available (fastest)
2. Load data in parallel from Hadoop cluster via SAS EP
3. Serial loads via SAS/Access ok for small tables


Any Supported SAS Serial Load
Host_L1
Host_L2
Host_Ln
SAS LASR Analytic Server
SAS High-Performance Analytics Environment Root Node
SAS High-Performance Analytics Environment Worker Node
SAS High-Performance Analytics Environment Worker Node
Serial Data Source
X


SASHDAT Parallel Load
Host_1
Host_2
Host_3
Host_n
SAS LASR Analytic Server
SAS High-Performance Analytics Environment Root Node
SAS High-Performance Analytics Environment Worker Node
SAS High-Performance Analytics Environment Worker Node
SAS High-Performance Analytics Environment Worker Node
Hadoop Name Node
Hadoop Data Node
SASHDAT
Hadoop Data Node
SASHDAT
Hadoop Data Node
SASHDAT
Hadoop Distributed File System
✓


SAS Embedded Process Parallel Load
Host_L1
Host_L2
Host_Li
SAS LASR Analytic Server
SAS High-Performance Analytics Environment Root Node
SAS High-Performance Analytics Environment Worker Node
SAS High-Performance Analytics Environment Worker Node
EP
Hadoop Name Node
EP
Hadoop Data Node
EP
Hadoop Data Node
Hadoop Distributed File System
Host_H1
Host_H2
Host_Hj
✓

THE SASHDAT LIBNAME STATEMENT



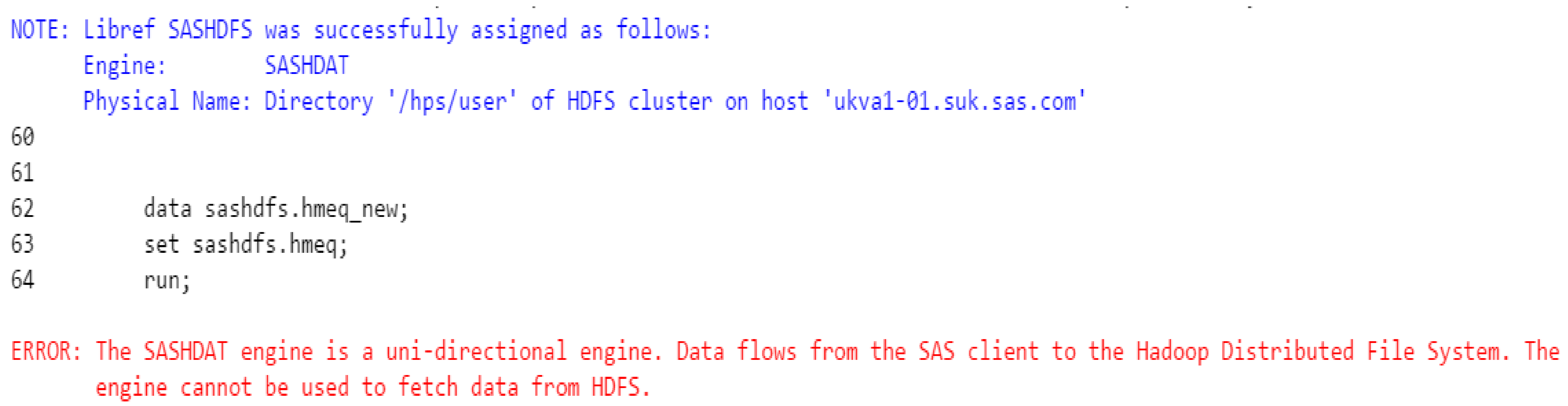
```
/*libname to local HDFS storing SASHDAT data*/
libname sashdfs sashdat path="/hps/user" HOST="xxxxxx-01.suk.sas.com" install="/opt/TKGrid";
```

option	notes
sashdat	The SAS engine which refers to HDFS
Path=	The hdfs path
Host=	The hostname of the TKGrid head node
Install=	The path where the SAS TKGrid binaries are installed

SASHDAT

SASHDAT IS A UNIDIRECTIONAL ENGINE

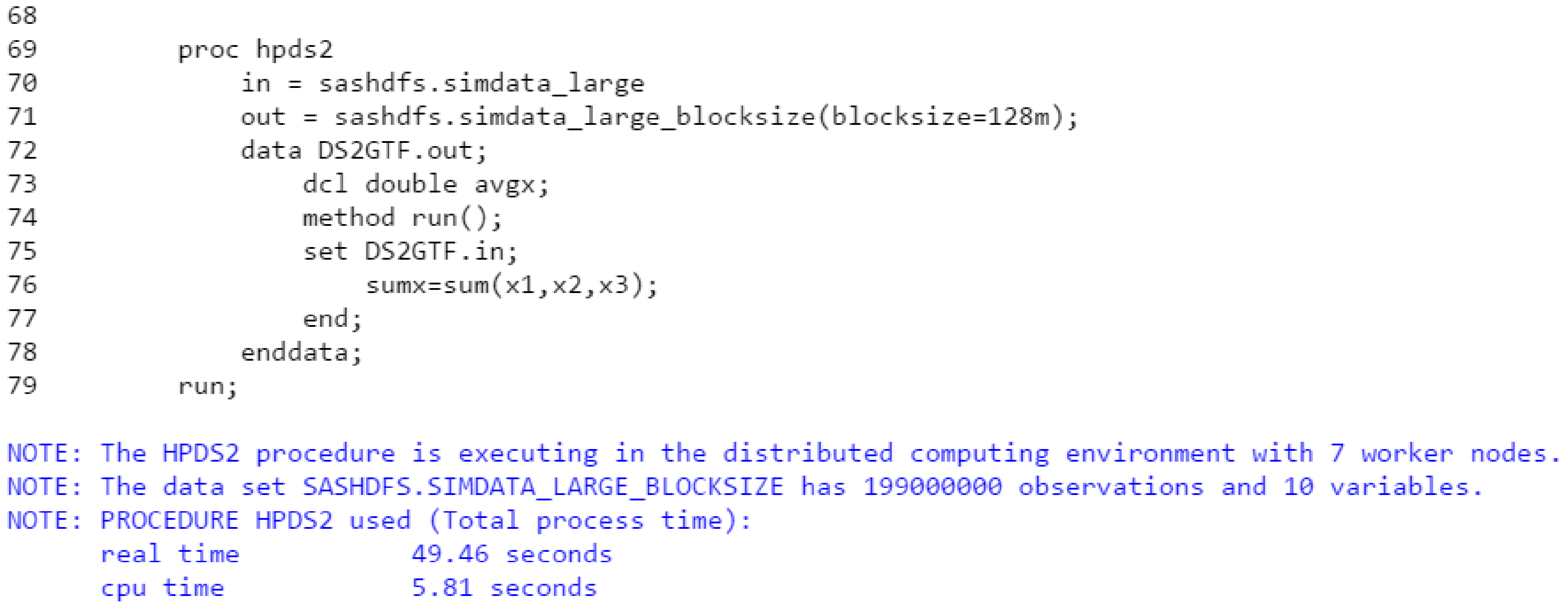
You can create data using the SASHDAT engine but you cannot re-read it. E.g.



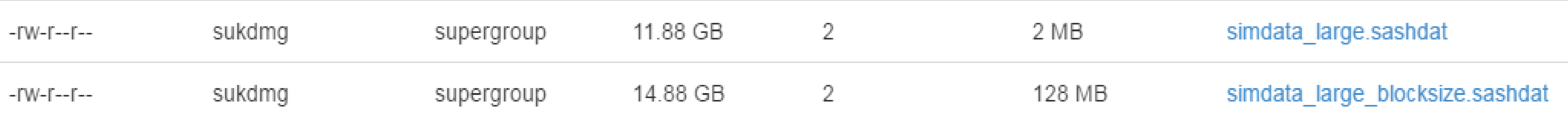
```
NOTE: Libref SASHDFS was successfully assigned as follows:
      Engine:        SASHDAT
      Physical Name: Directory '/hps/user' of HDFS cluster on host 'ukva1-01.suk.sas.com'
60
61
62           data sashdfs.hmeq_new;
63           set sashdfs.hmeq;
64           run;

ERROR: The SASHDAT engine is a uni-directional engine. Data flows from the SAS client to the Hadoop Distributed File System. The
       engine cannot be used to fetch data from HDFS.
```

SASHDAT USE PROC HPDS2 TO MANIPULATE SASHDAT DATA

Proc HPDS2 can be used to create a new sashdat files from a sashdat file



```
      proc hpds2
          in = sashdfs.simdata_large
          out = sashdfs.simdata_large_blocksize(blocksize=128m);
          data DS2GTF.out;
              dcl double avgx;
              method run();
              set DS2GTF.in;
                  sumx=sum(x1,x2,x3);
              end;
          enddata;
      run;

NOTE: The HPDS2 procedure is executing in the distributed computing environment with 7 worker nodes.
NOTE: The data set SASHDFS.SIMDATA_LARGE_BLOCKSIZE has 199000000 observations and 10 variables.
NOTE: PROCEDURE HPDS2 used (Total process time):
      real time           49.46 seconds
      cpu time            5.81 seconds
```

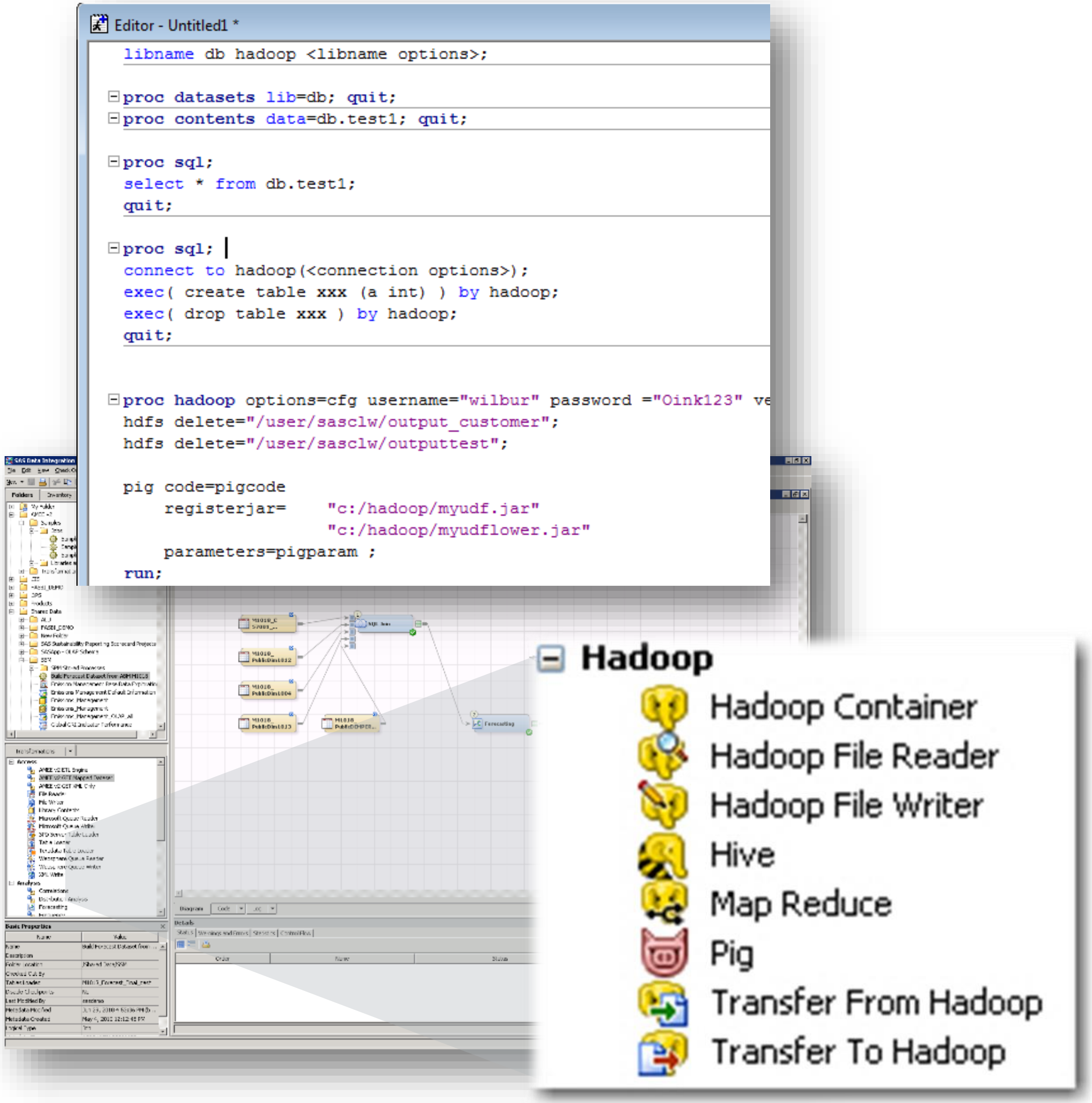


-rw-r--r--	sukdmg	supergroup	11.88 GB	2	2 MB	simdata_large.sashdat
-rw-r--r--	sukdmg	supergroup	14.88 GB	2	128 MB	simdata_large_blocksize.sashdat

SAS/ACCESS AND SPDE ON HDFS

SAS/ACCESS® TO HADOOP

- Uses Existing SAS Interfaces
- Standard Libname syntax
- PROC HADOOP
- Datastep and Proc SQL translated to Hive
- Filename support
- Execute Pig Scripts and MapReduce
- Push-down of certain procedures
- Custom SerDe support
- SPDE formats



Editor - Untitled1 *

```
libname db hadoop <libname options>;

proc datasets lib=db; quit;
proc contents data=db.test1; quit;

proc sql;
select * from db.test1;
quit;

proc sql;
connect to hadoop(<connection options>);
exec( create table xxx (a int) ) by hadoop;
exec( drop table xxx ) by hadoop;
quit;


proc hadoop options=cfg username="wilbur" password ="Oink123" ve
hdfs delete="/user/sasclw/output_customer";
hdfs delete="/user/sasclw/outputtest";

pig code=pigcode
    registerjar=    "c:/hadoop/myudf.jar"
                    "c:/hadoop/myudflower.jar"
    parameters=pigparam ;
run;
```

SAS/ACCESS TO HADOOP

- HIVE Data types (avoid strings, use VARCHAR for character fields)
- Use native Hadoop file formats (ORC, PARQUET etc.) and partition data where appropriate
- Make use of supported In-database SAS procedures
  - FREQ, MEANS, REPORT, SUMMARY/MEANS, TABULATE

Data integration:

- Use the standard SQL transformations in DI
- Generate explicit pass-through
- Create and manage SASHDAT and LASR tables using the DI transformations

MAKING USE OF YARN QUEUES

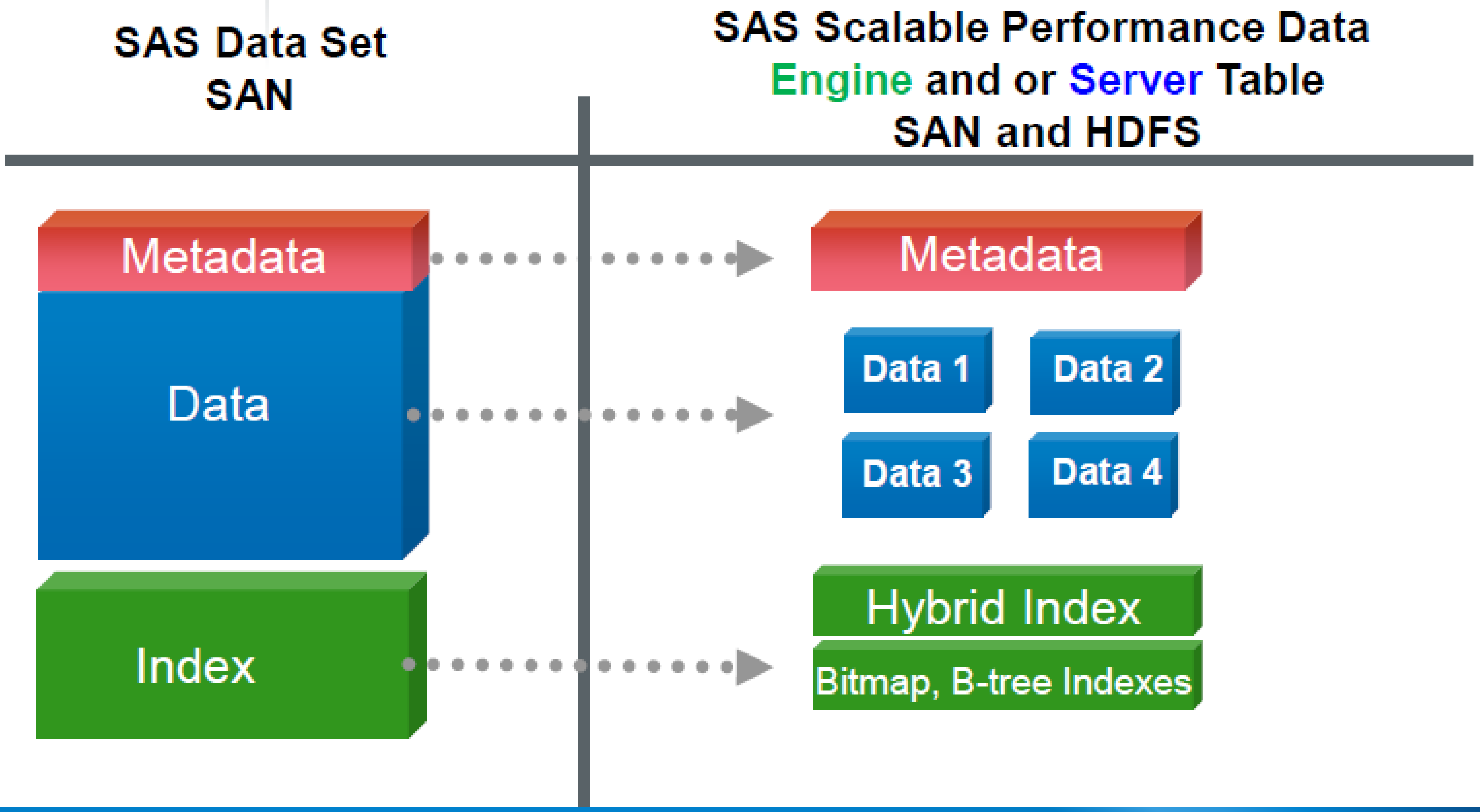
- Setting Hive Queue:
  - PROPERTIES= option can be added to the LIBNAME statement to add properties, like mapreduce.job.queuename, to the library connection. (http://support.sas.com/documentation/cdl/en/acreldb/68028/HTML/default/viewer.htm#p0ly2onqqpbys8n1j9lra8qa6q20.htm)

```
Libname hivetez hadoop server="gbrhadoop1-01" USER=sasdemo
PASSWORD="{SAS002}1D57933958C580064BD3DCA81A33DFB2"
port=10000 PROPERTIES='mapreduce.job.queuename=sas_user_queue‘
DBCREATE_TABLE_OPTS='STORED AS PARQUET';
```

PUSH DOWN THE SQL PROCESSING TO HIVE AS MUCH AS POSSIBLE

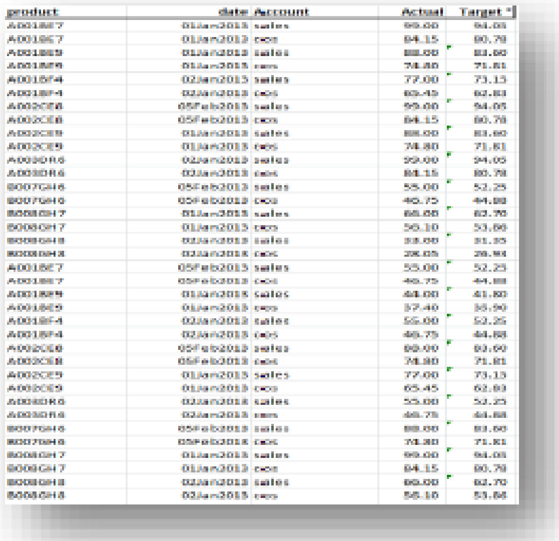
- Avoid joining SAS data with Hive Data. It is recommended to move the SAS dataset into Hive and execute the join inside Hadoop to leverage distributed processing
- Avoid using SAS functions that will bring back Hadoop data on the SAS Server because the function does not exist in HIVE. E.g. datepart
- Use SASTRACE option to see the communication between SAS and Hadoop.

SPDE ON HDFS

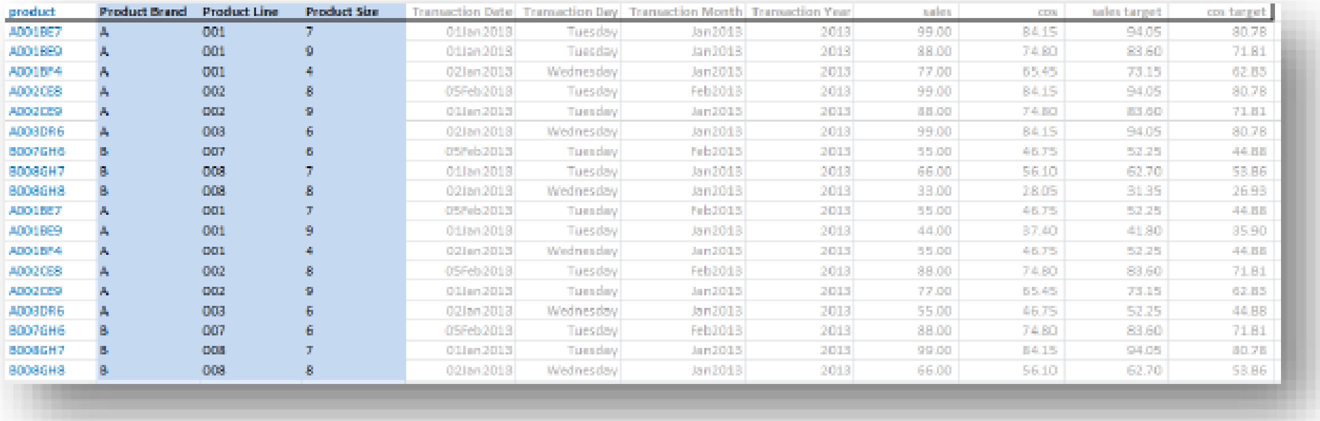

SAS Data Set
SAN
SAS Scalable Performance Data
Engine and or Server Table
SAN and HDFS
Metadata
Data
Index
Metadata
Data 1
Data 2
Data 3
Data 4
Hybrid Index
Bitmap, B-tree Indexes

SPDE ON HDFS

Analytical Base Table



product	date	Account	Actual	Target
A001BE7	01Jan2013	sales	99.00	94.05
A001BE7	01Jan2013	cos	84.15	80.78
A001BE9	01Jan2013	sales	88.00	83.60
A001BE9	01Jan2013	cos	74.80	71.81
A001BF4	02Jan2013	sales	77.00	73.15
A001BF4	02Jan2013	cos	65.45	62.83
A002CE8	05Feb2013	sales	99.00	94.05
A002CE8	05Feb2013	cos	84.15	80.78
A002CE9	01Jan2013	sales	88.00	83.60
A002CE9	01Jan2013	cos	74.80	71.81
A003DR6	02Jan2013	sales	99.00	94.05
A003DR6	02Jan2013	cos	84.15	80.78
B007GH6	05Feb2013	sales	55.00	52.25
B007GH6	05Feb2013	cos	46.75	44.88
B008GH7	01Jan2013	sales	66.00	62.70
B008GH7	01Jan2013	cos	56.10	53.86
B008GH8	02Jan2013	sales	33.00	31.35
B008GH8	02Jan2013	cos	28.05	26.93
A001BE7	05Feb2013	sales	55.00	52.25
A001BE7	05Feb2013	cos	46.75	44.88
A001BE9	01Jan2013	sales	44.00	41.80
A001BE9	01Jan2013	cos	37.40	35.90
A001BF4	02Jan2013	sales	55.00	52.25
A001BF4	02Jan2013	cos	46.75	44.88
A002CE8	05Feb2013	sales	88.00	83.60
A002CE8	05Feb2013	cos	74.80	71.81
A002CE9	01Jan2013	sales	77.00	73.15
A002CE9	01Jan2013	cos	65.45	62.83
A003DR6	02Jan2013	sales	55.00	52.25
A003DR6	02Jan2013	cos	46.75	44.88
B007GH6	05Feb2013	sales	88.00	83.60
B007GH6	05Feb2013	cos	74.80	71.81
B008GH7	01Jan2013	sales	99.00	94.05
B008GH7	01Jan2013	cos	84.15	80.78
B008GH8	02Jan2013	sales	66.00	62.70
B008GH8	02Jan2013	cos	56.10	53.86



product	Product Brand	Product Line	Product Size	Transaction Date	Transaction Day	Transaction Month	Transaction Year	sales	cos	sales target	cos target
A001BE7	A	001	7	01Jan2013	Tuesday	Jan2013	2013	99.00	84.15	94.05	80.78
A001BE9	A	001	9	01Jan2013	Tuesday	Jan2013	2013	88.00	74.80	83.60	71.81
A001BF4	A	001	4	02Jan2013	Wednesday	Jan2013	2013	77.00	65.45	73.15	62.83
A002CE8	A	002	8	05Feb2013	Tuesday	Feb2013	2013	99.00	84.15	94.05	80.78
A002CE9	A	002	9	01Jan2013	Tuesday	Jan2013	2013	88.00	74.80	83.60	71.81
A003DR6	A	003	6	02Jan2013	Wednesday	Jan2013	2013	99.00	84.15	94.05	80.78
B007GH6	B	007	6	05Feb2013	Tuesday	Feb2013	2013	55.00	46.75	52.25	44.88
B008GH7	B	008	7	01Jan2013	Tuesday	Jan2013	2013	66.00	56.10	62.70	53.86
B008GH8	B	008	8	02Jan2013	Wednesday	Jan2013	2013	33.00	28.05	31.35	26.93
A001BE7	A	001	7	05Feb2013	Tuesday	Feb2013	2013	55.00	46.75	52.25	44.88
A001BE9	A	001	9	01Jan2013	Tuesday	Jan2013	2013	44.00	37.40	41.80	35.90
A001BF4	A	001	4	02Jan2013	Wednesday	Jan2013	2013	55.00	46.75	52.25	44.88
A002CE8	A	002	8	05Feb2013	Tuesday	Feb2013	2013	88.00	74.80	83.60	71.81
A002CE9	A	002	9	01Jan2013	Tuesday	Jan2013	2013	77.00	65.45	73.15	62.83
A003DR6	A	003	6	02Jan2013	Wednesday	Jan2013	2013	55.00	46.75	52.25	44.88
B007GH6	B	007	6	05Feb2013	Tuesday	Feb2013	2013	88.00	74.80	83.60	71.81
B008GH7	B	008	7	01Jan2013	Tuesday	Jan2013	2013	99.00	84.15	94.05	80.78
B008GH8	B	008	8	02Jan2013	Wednesday	Jan2013	2013	66.00	56.10	62.70	53.86

Meant to support VERY wide tables for Predictive Analytics, Visualization, Dashboards, Self Service Reporting

SPDE ON HDFS

Can sometimes be faster than HIVE access when working with SAS :

- Depending on the queries (no need to deal with Hive, direct access via HDFS)
- Can be faster than HIVE when used as input to SAS HPA procedures

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

SAS PROGRAMMERS LEVERAGING HADOOP USING SPD ENGINE

1. Use **PROC HADOOP** to create the path on HDFS:

```
proc hadoop
   username='Hadoop_userid`
   password='Hadoop_password'
verbose;
   hdfs mkdir='/user/sasss1/spde';
run;
```

2. **SPD Engine** LIBNAME statement:

```
LIBNAME MYSPDE SPDE
'/user/sasss1/spde’
HDFSHOST=DEFAULT
PARALLELWRITE=YES
PARALLELREAD=YES
ACCELWHERE=YES;
```

1. *MYSPDE* is the libref we reference in our SAS code to process the SPD Engine data stored on HDFS.
2. *SPDE* is the engine SPD Engine uses to process SPD Engine tables.
3. **'/user/sasss1/spde' is the path on HDFS where our SPD Engine data is stored.**
4. *HDFSHOST*=DEFAULT To connect to the Hadoop cluster, Hadoop configuration files must be copied from the specific Hadoop cluster to a physical location that the SAS client machine can access. The SAS environment variable SAS_HADOOP_CONFIG_PATH must be defined and set to the location of the Hadoop configuration files. For complete instructions, see the SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS.
5. *PARALLELWRITE*=YES tells SPD Engine to use parallel processing to write data to HDFS. **Note: data must be uncompressed.**
6. *PARALLELREAD*=YES tells SPD Engine to use parallel processing to read data stored in HDFS. Note: data can be uncompressed, compressed or encrypted.
7. *ACCELWHERE*=YES tells SPD Engine, when possible, to push all WHERE clauses down to Hadoop as MapReduce

HADOOP AND SAS FILE FORMATS

Engine	Typically, use for	3rd party Hadoop access
HIVE (ORC - HDP) (Parquet - CDH) (AVRO)	• Data that needs to be available for processing by the broader Hadoop ecosystem • Data to be processed by pushdown SQL queries or SAS DS2.	yes
SASHDAT (only supported for symmetric deployments)	• Persisting data on HDFS and for the fast, parallel loading of data into LASR/HPA	No
SPDE on HDFS	• Migrating SAS data sets onto HDFS without code modification • Optimised data retrieval back to SAS. • Input to LASR/HPA (faster than HIVE) • Very wide analytical base tables	Yes – read only access via SAS supplied SerDe

OVERVIEW OF PROCESSING OPTIONS

SAS Programming Method	SPDE	HIVE	SASHDAT	LASR (SASIOLA)
Proc SQL implicit	Yes	Yes – via SAS/Access	No	No*
Proc SQL explicit	No	Yes – via SAS/Access	No	No
Data Step	Yes - via SAS EP	Yes - via SAS EP	No**	Yes
Proc DS2	Yes - via SAS EP	Yes - via SAS EP	No	No
Proc HPDS2	Yes	No	Yes	Yes

*Would work but will pull data to SAS client for processing
**Can be used to create new SASHDAT datasets but not to modify data

SAS EMBEDDED PROCESS AND DS2

THE SAS EMBEDDED PROCESS: A WORD ON THE TECHNOLOGY

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

```
proc ds2 ;
/* thread ~ eqiv to a mapper */
   thread map_program;
   method run(); set dbmslib.intab;
   /* program sta      s */
   end; endthre
   /* program w
   data hdf.dat
dcl thread map_p      ap_pgm; method
run();
set from map_pgm threads=N;
/* reduce steps */ end; enddata;
run; quit;
```

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

RUN FASTER. RUN EMBEDDED

- Efficient way to process data.
- Runs inside Hadoop’s MPP architecture.
- Moves the computation to the data.
- Eliminates data movement.
- Decreases overall processing times.

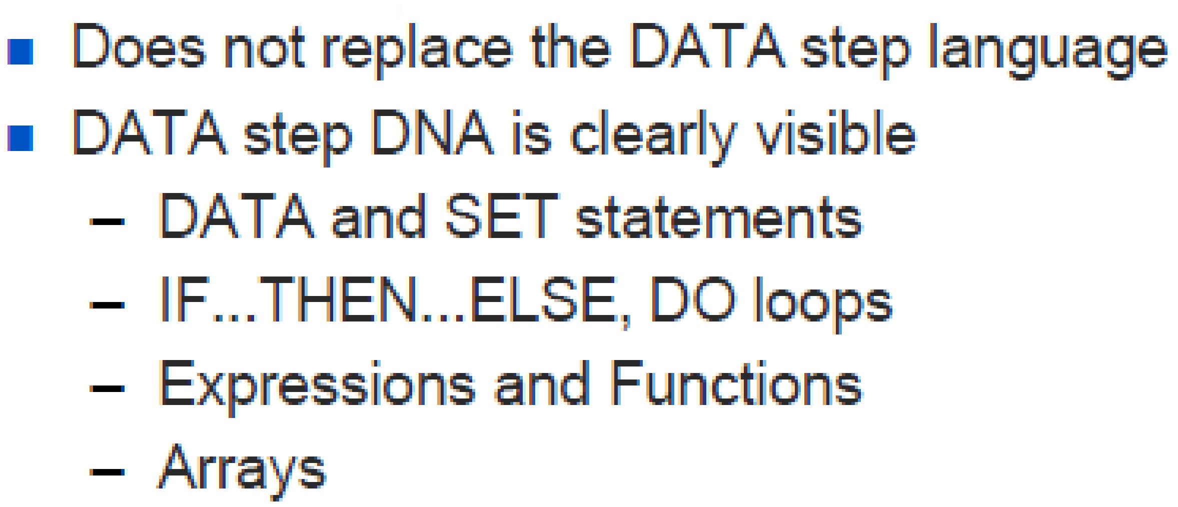
DS2 IN 30+ SECONDS

- Procedural programming language.
- Mainly focused around parallel execution.
- Supports ANSI SQL data types.
- Allows Embedded SQL as input to the program.
- Allows modular programming: Scope and Methods.
- Supports Packages and Threads.

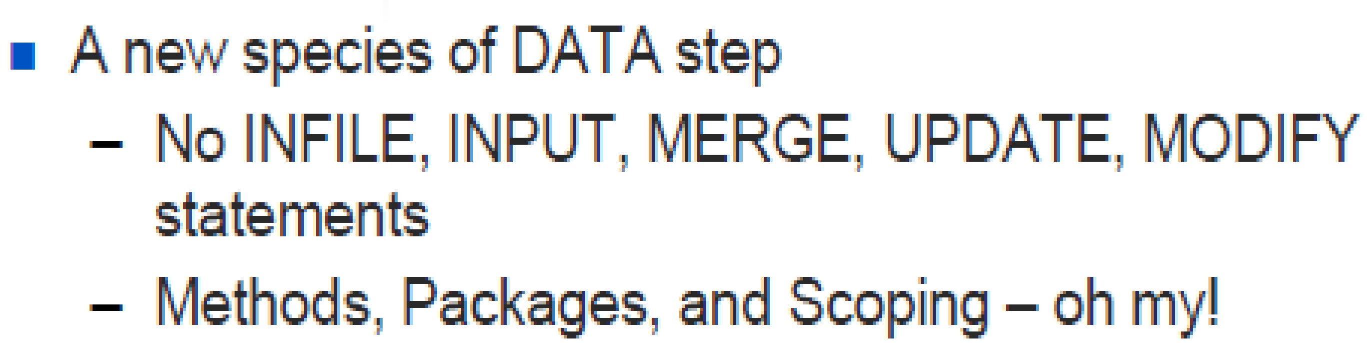
SAS EMBEDDED PROCESS FOR HADOOP

- Lightweight execution container for DS2.
- Written in C and Java.
- Runs inside a MapReduce task.
- Orchestrated by Hadoop MapReduce framework.
- Resource allocation managed by YARN.

WHAT IS DS2?

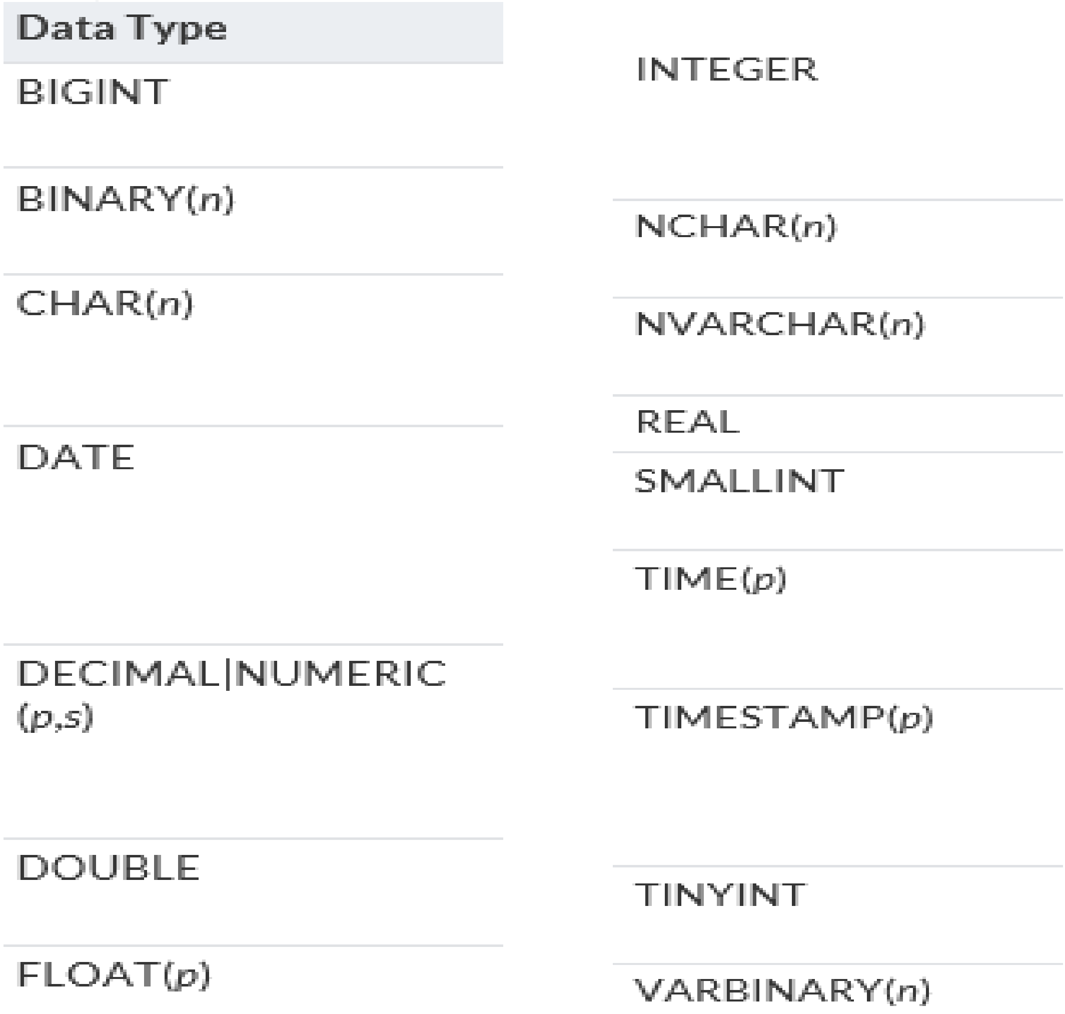


- Does not replace the DATA step language
- DATA step DNA is clearly visible
  - DATA and SET statements
  - IF...THEN...ELSE, DO loops
  - Expressions and Functions
  - Arrays



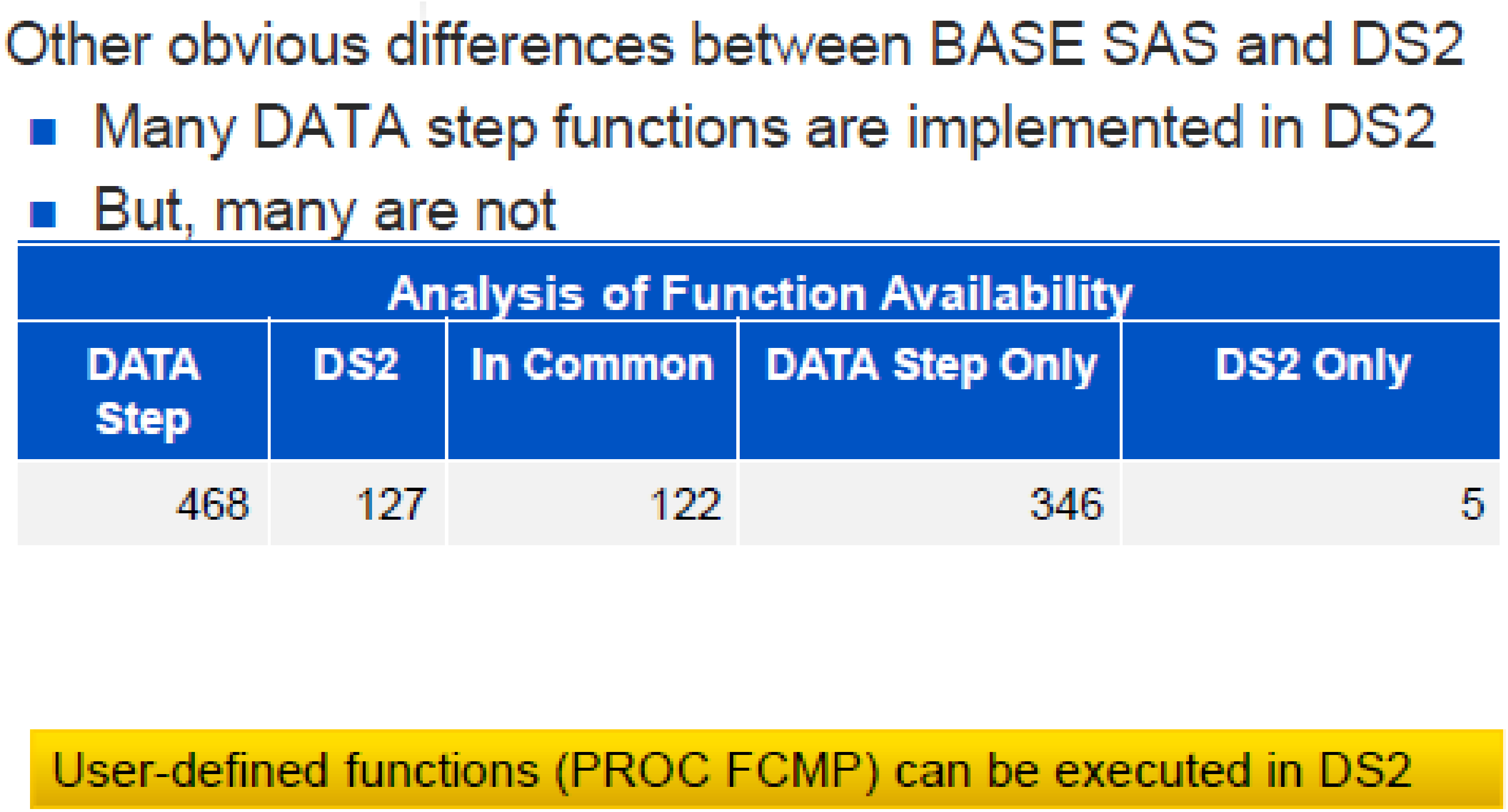
- A new species of DATA step
  - No INFILE, INPUT, MERGE, UPDATE, MODIFY statements
  - Methods, Packages, and Scoping – oh my!

DS2 WHAT IS DS2? DATA TYPES



Data Type
BIGINT
BINARY(*n*)
CHAR(*n*)
DATE
DECIMAL|NUMERIC (*p*,*s*)
DOUBLE
FLOAT(*p*)
INTEGER
NCHAR(*n*)
NVARCHAR(*n*)
REAL
SMALLINT
TIME(*p*)
TIMESTAMP(*p*)
TINYINT
VARBINARY(*n*)

DS2 DATA STEP SIMILARITIES/DIFFERENCES



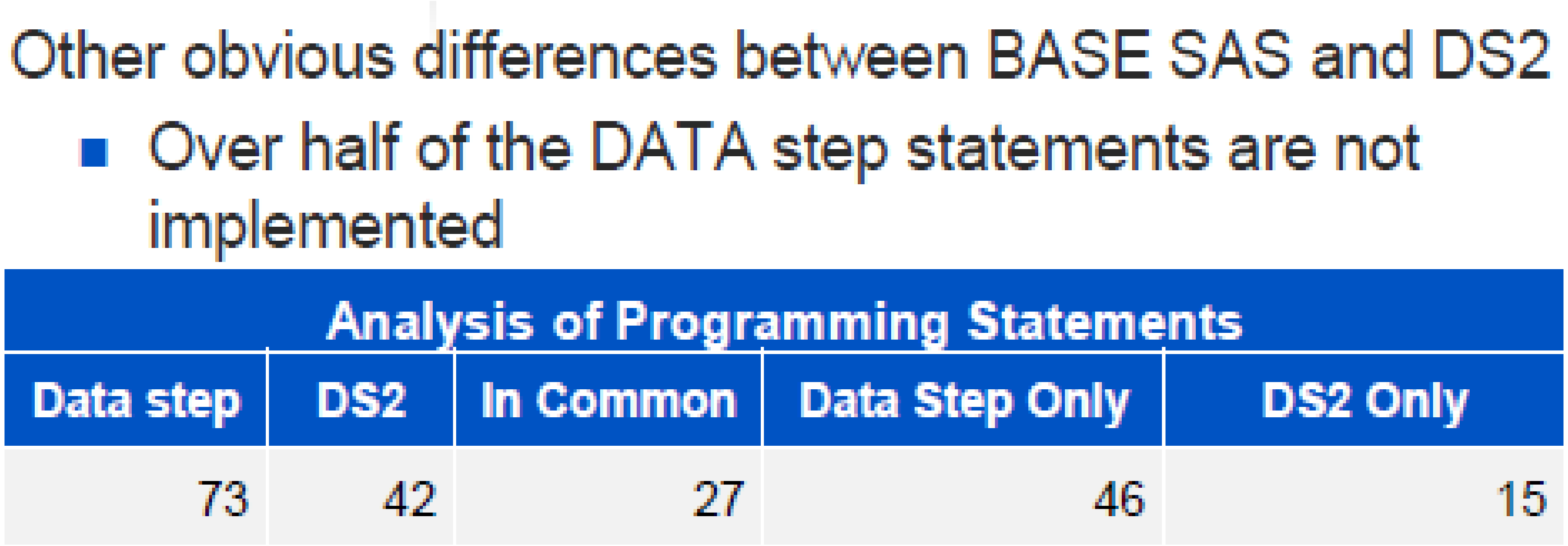
Other obvious differences between BASE SAS and DS2

- Many DATA step functions are implemented in DS2
- But, many are not

Analysis of Function Availability				
DATA Step	DS2	In Common	DATA Step Only	DS2 Only
468	127	122	346	5

User-defined functions (PROC FCMP) can be executed in DS2

DS2 DATA STEP SIMILARITIES/DIFFERENCES



Other obvious differences between BASE SAS and DS2

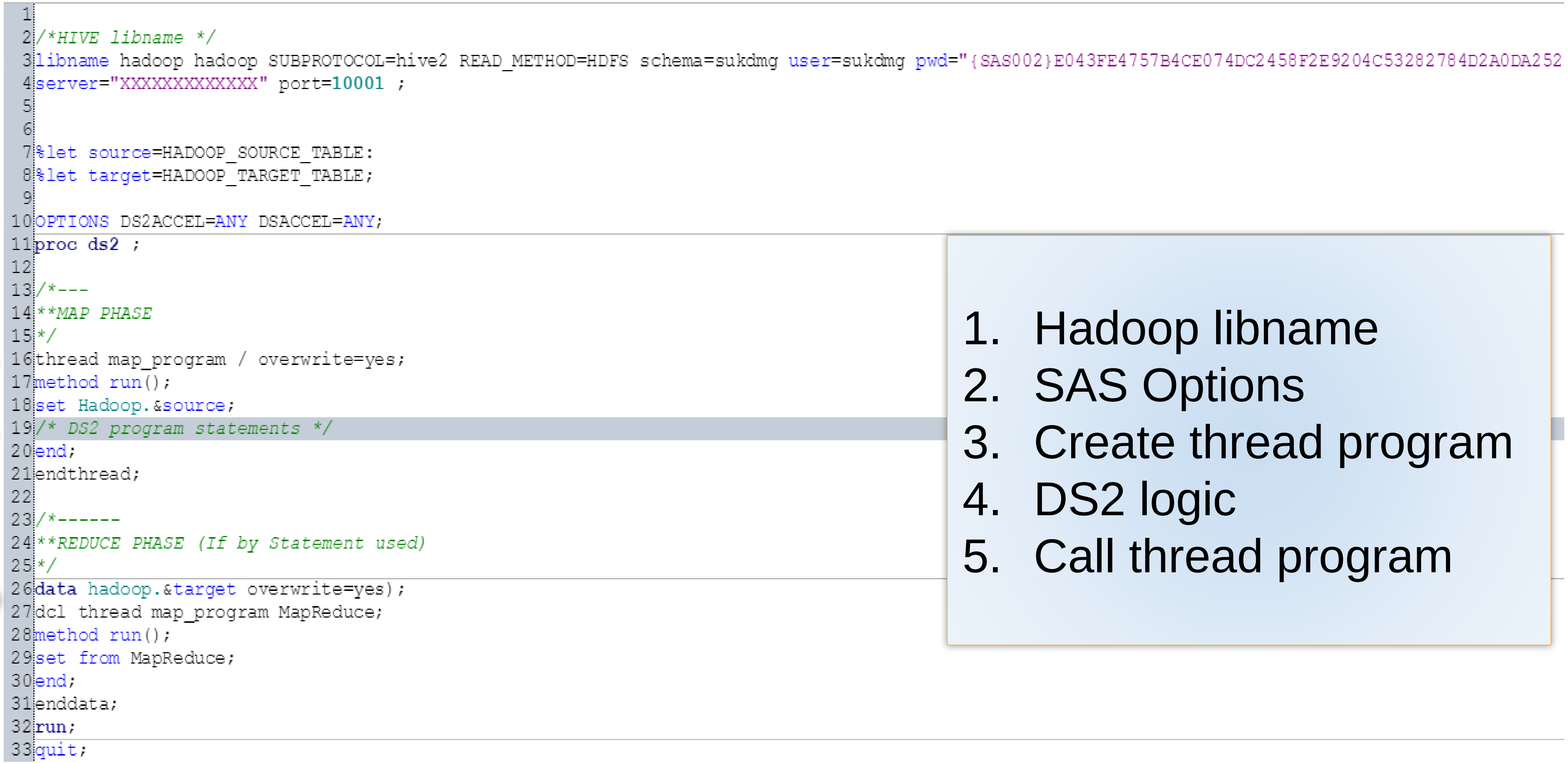
- Over half of the DATA step statements are not implemented

Analysis of Programming Statements				
Data step	DS2	In Common	Data Step Only	DS2 Only
73	42	27	46	15

RUNNING SAS DATA STEP & DS2 IN HADOOP THROUGH THE CODE ACCELERATOR

- Key SAS options: DSACCELL=ANY and DS2ACCELL=ANY
- DS2 in Hadoop supports both HIVE and SAS SPDE tables
  - Use proc HPDS2 to manipulate SASHDAT tables

DS2 SYNTAX FRAMEWORK FOR HADOOP

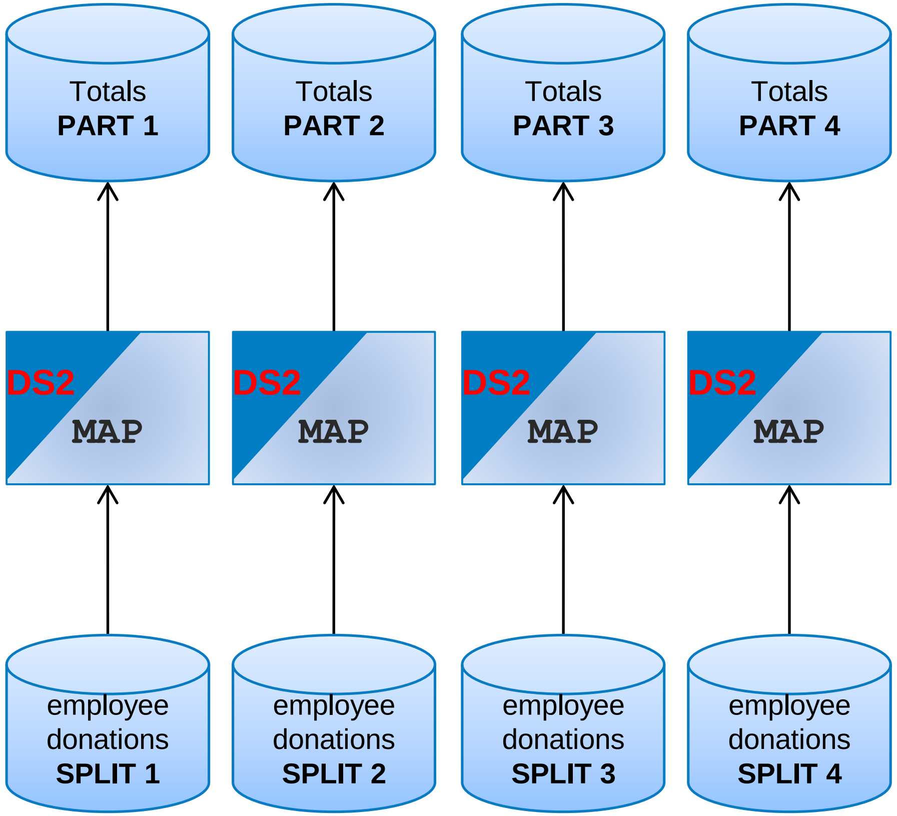


```

/*HIVE libname */
libname hadoop hadoop SUBPROTOCOL=hive2 READ_METHOD=HDFS schema=sukdmg user=sukdmg pwd="{SAS002}E043FE4757B4CE074DC2458F2E9204C53282784D2A0DA252
server="XXXXXXXXXXXX" port=10001 ;


%let source=HADOOP_SOURCE_TABLE:
%let target=HADOOP_TARGET_TABLE;

OPTIONS DS2ACCEL=ANY DSACCEL=ANY;
proc ds2 ;

/*---
**MAP PHASE
*/
thread map_program / overwrite=yes;
method run();
set Hadoop.&source;
/* DS2 program statements */
end;
endthread;

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

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

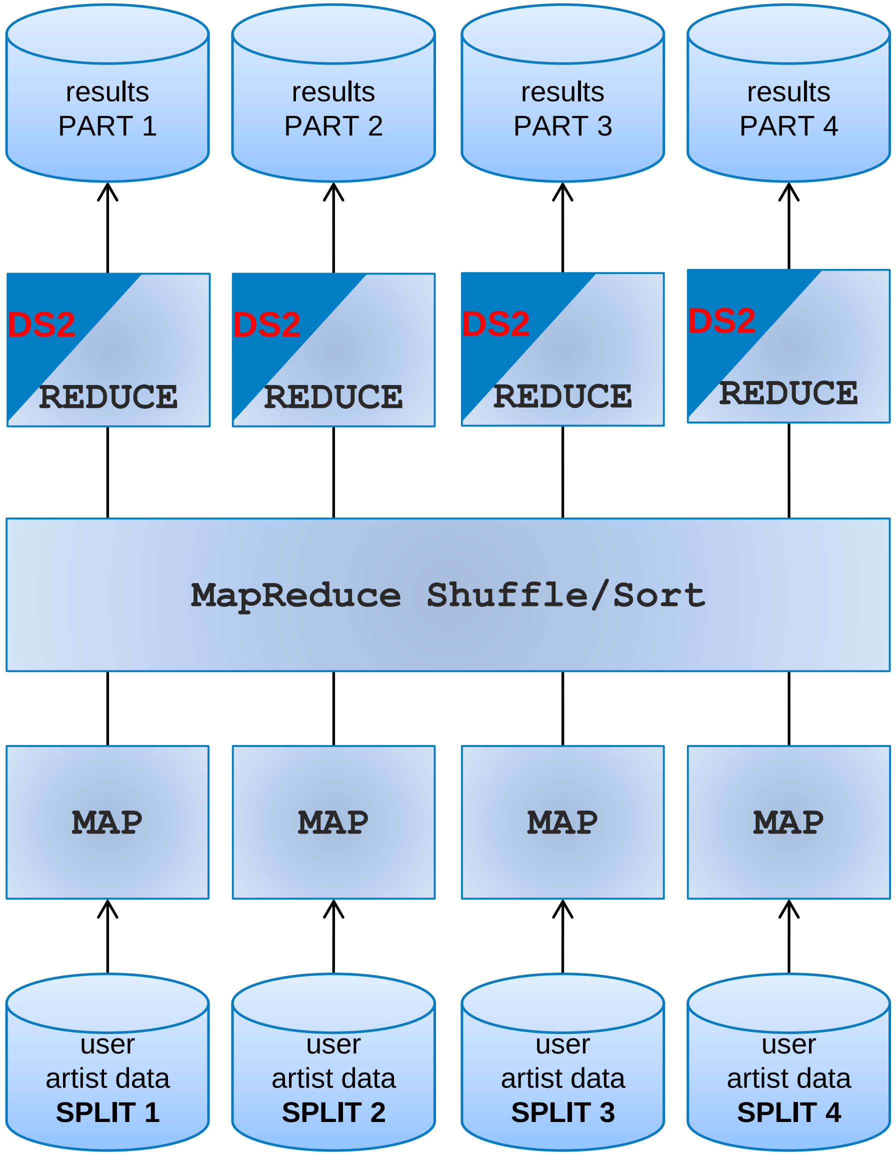
DS2 IN HADOOP WITH CODE ACCELERATOR

```
proc ds2 ds2accel=yes;
thread compute;
  method run();
    set hdfs.emp_donations;
    total = sum(jan--dec);
  end;
endthread;
data hdfs.totals;
  dcl thread compute t;
  method run();
    set from t;
  end;
enddata;
run; quit;
```

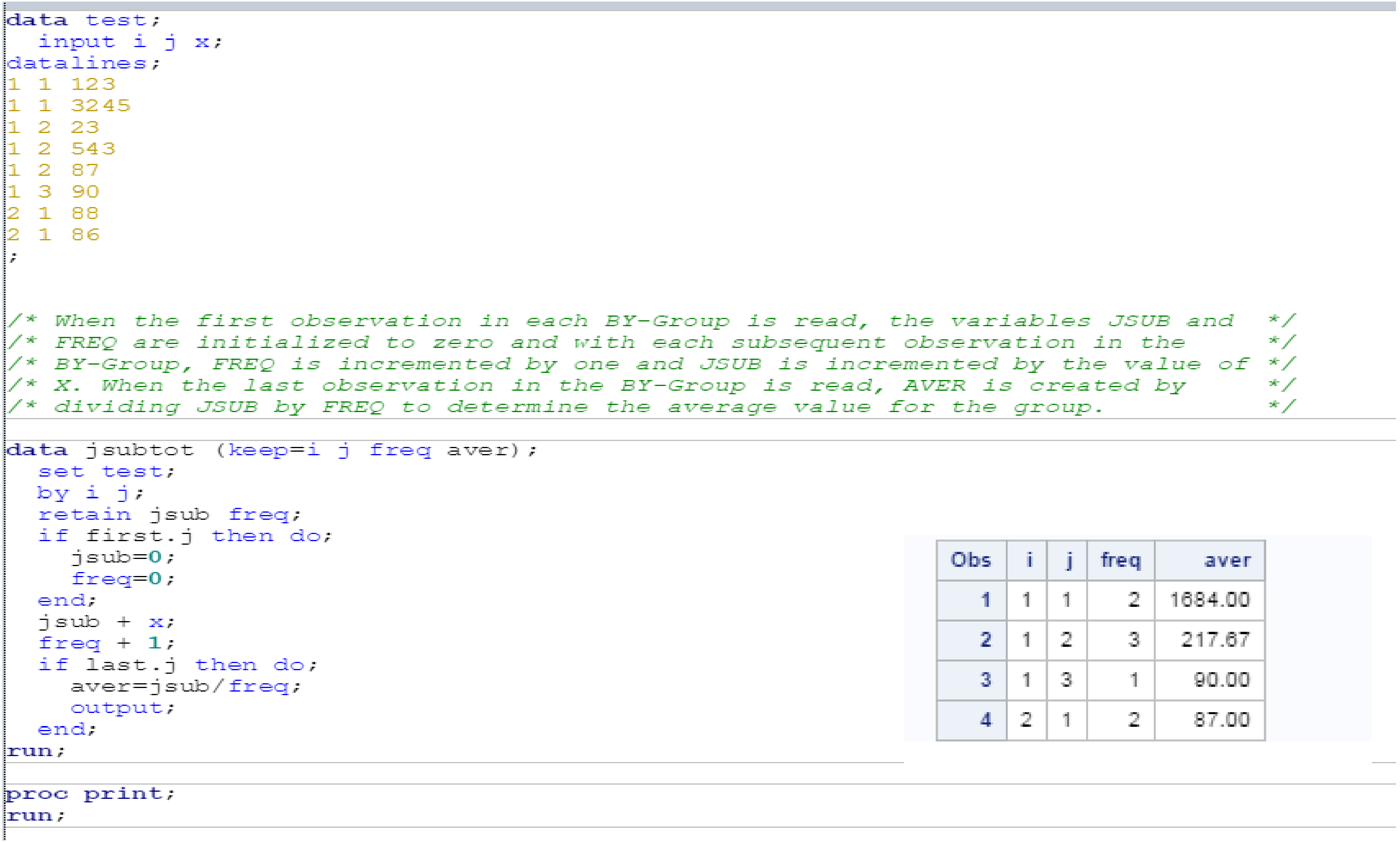


DS2 IN HADOOP WITH BY GROUP PROCESSING

```
proc ds2 ds2accel=yes;
thread compute;
  method run();
    set hdfs.emp_donations;
    by region;
    if first.region then total = 0;
    total + sum(jan--dec);
    if last.region then output;
  end;
endthread;
data hdfs.totals;
  dcl thread compute t;
  method run();
    set from t;
  end;
enddata;
```



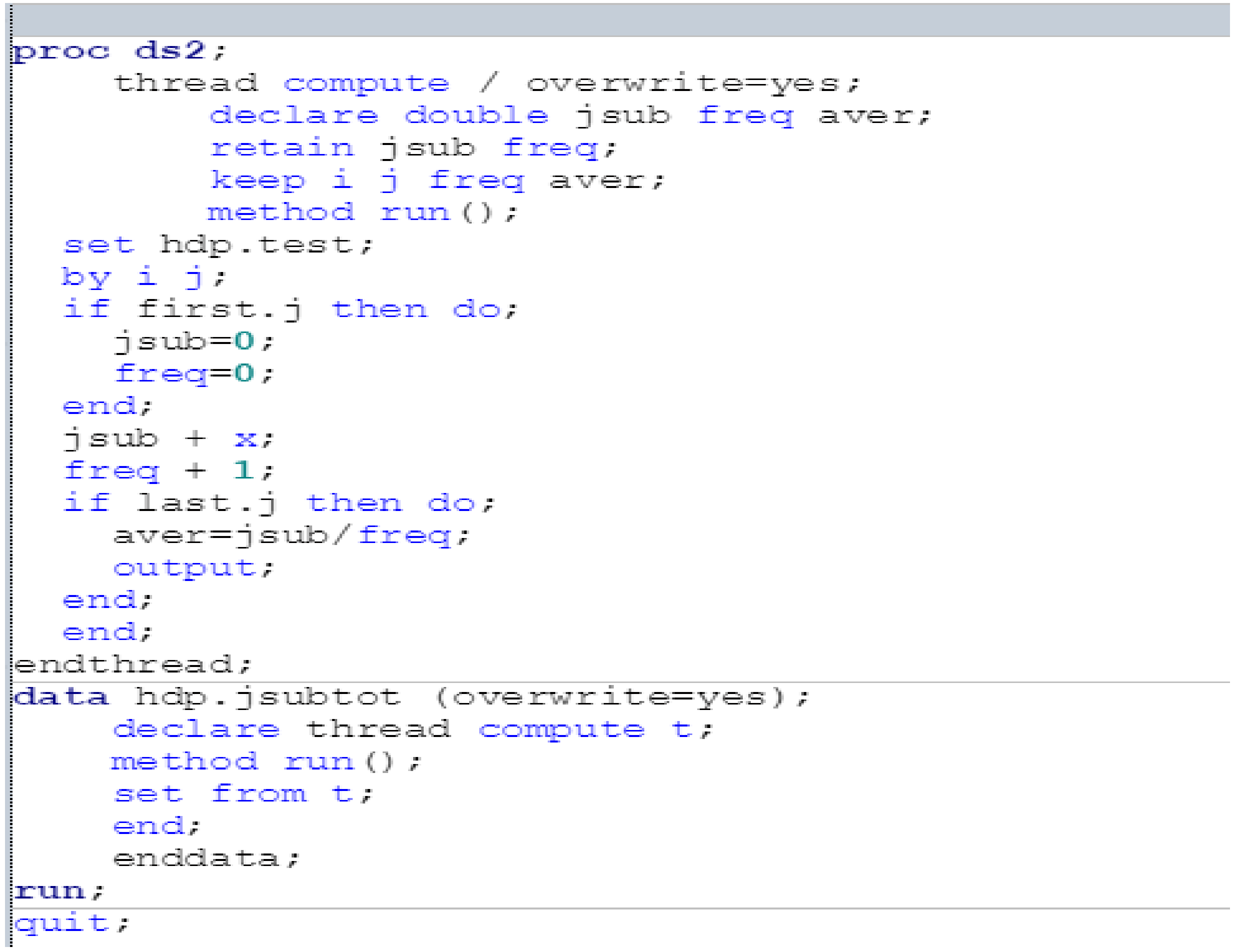
ORIGINAL DATA STEP PROGRAM



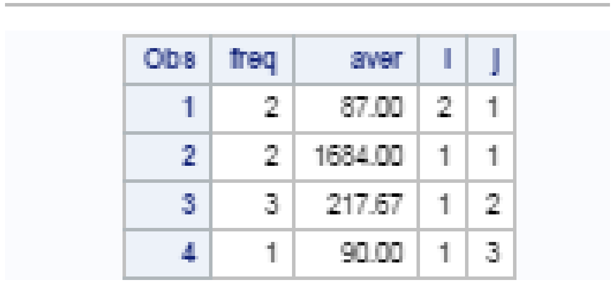
```
data test;
  input i j x;
datalines;
1 1 123
1 1 3245
1 2 23
1 2 543
1 2 87
1 3 90
2 1 88
2 1 86
;


/* 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

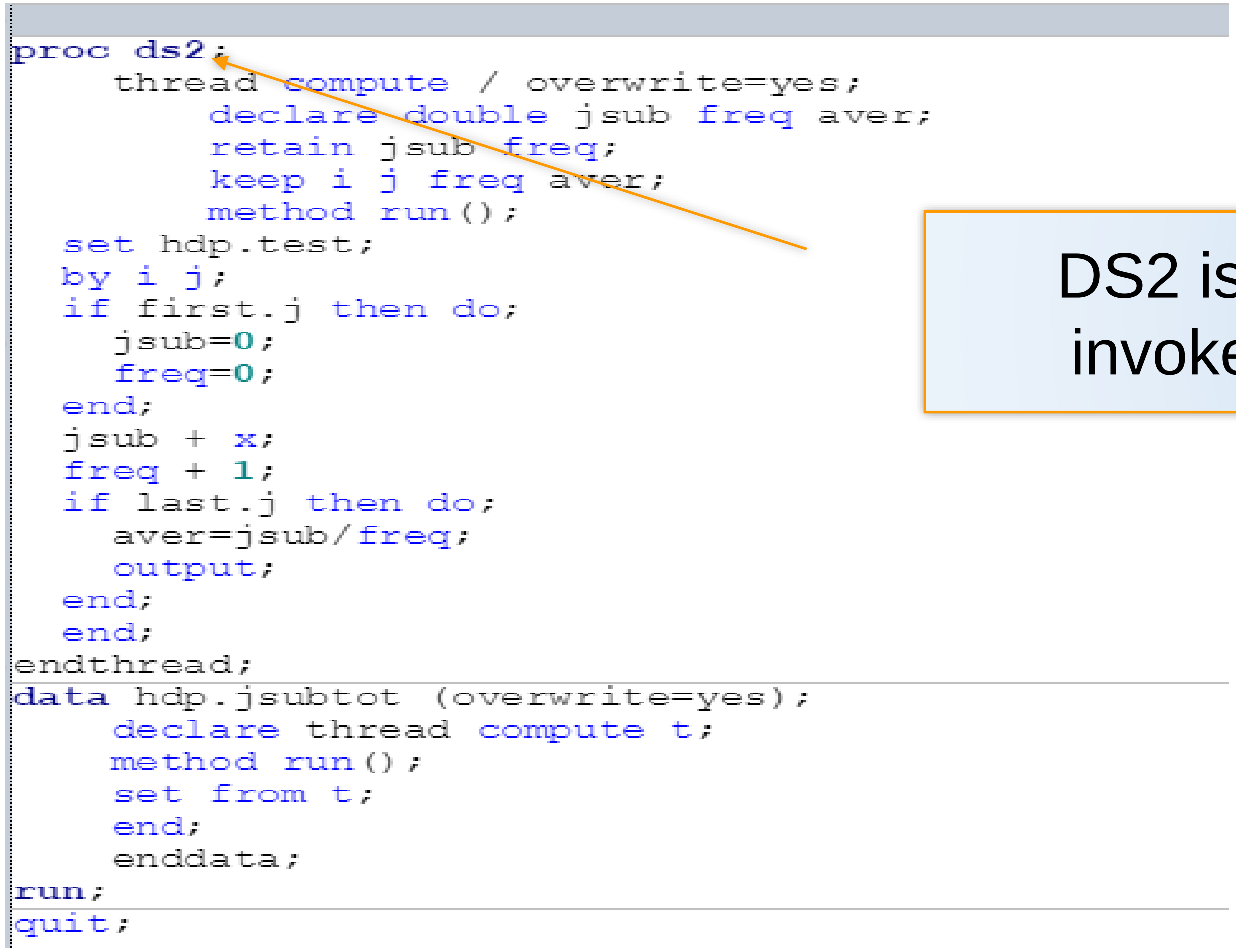


```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;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

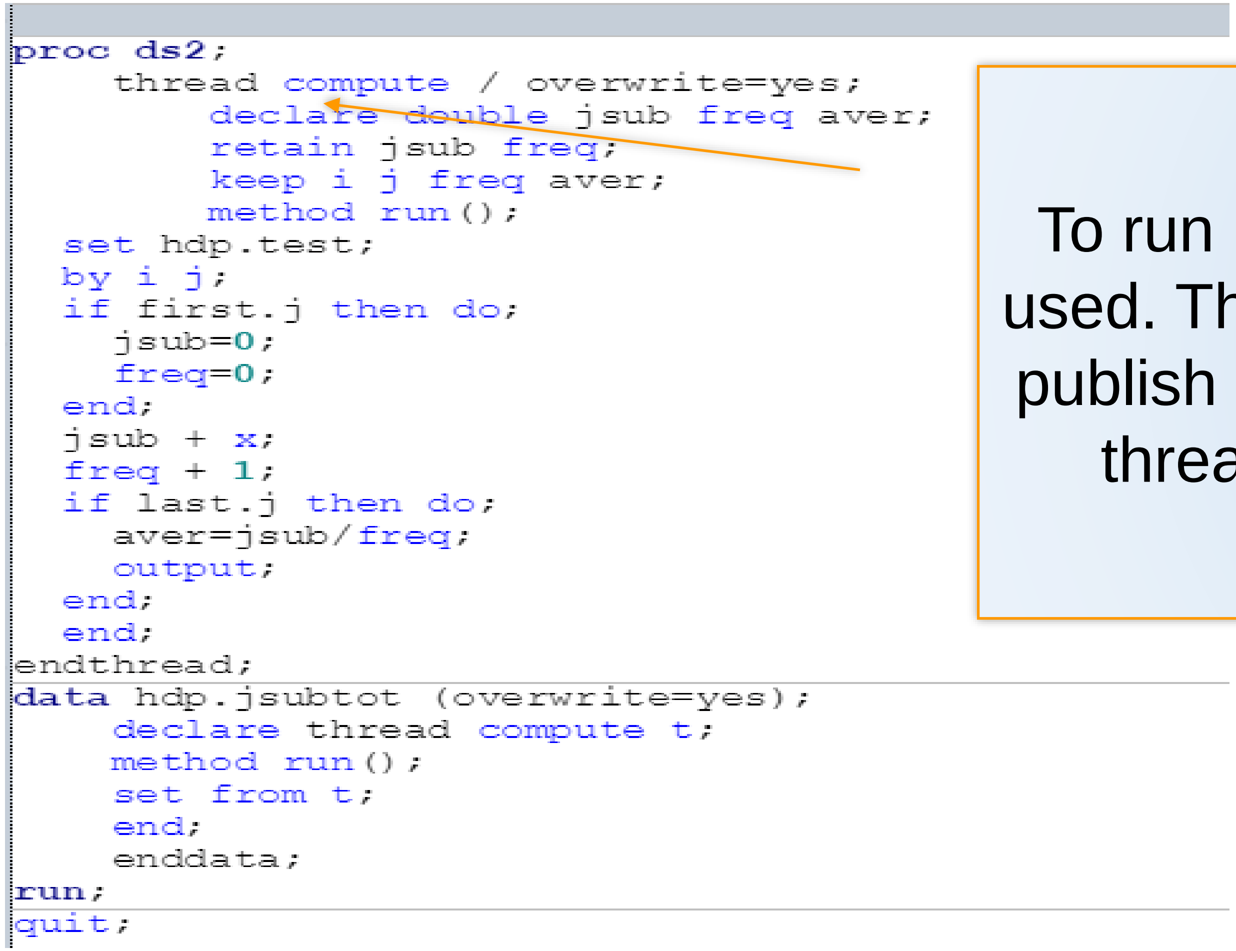
DS2 EQUIVALENT FOR HADOOP



```
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.

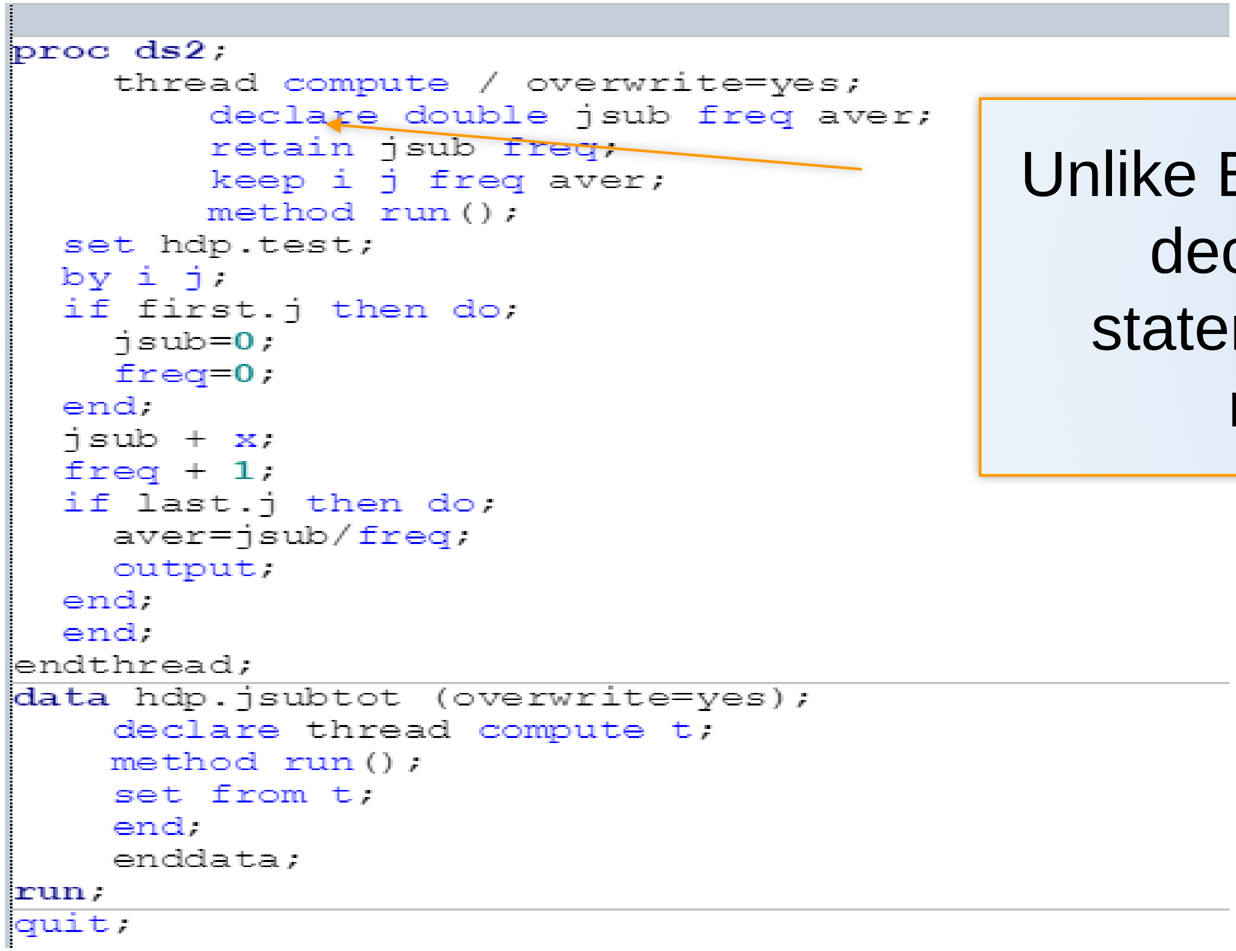
DS2 EQUIVALENT FOR HADOOP



```
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.

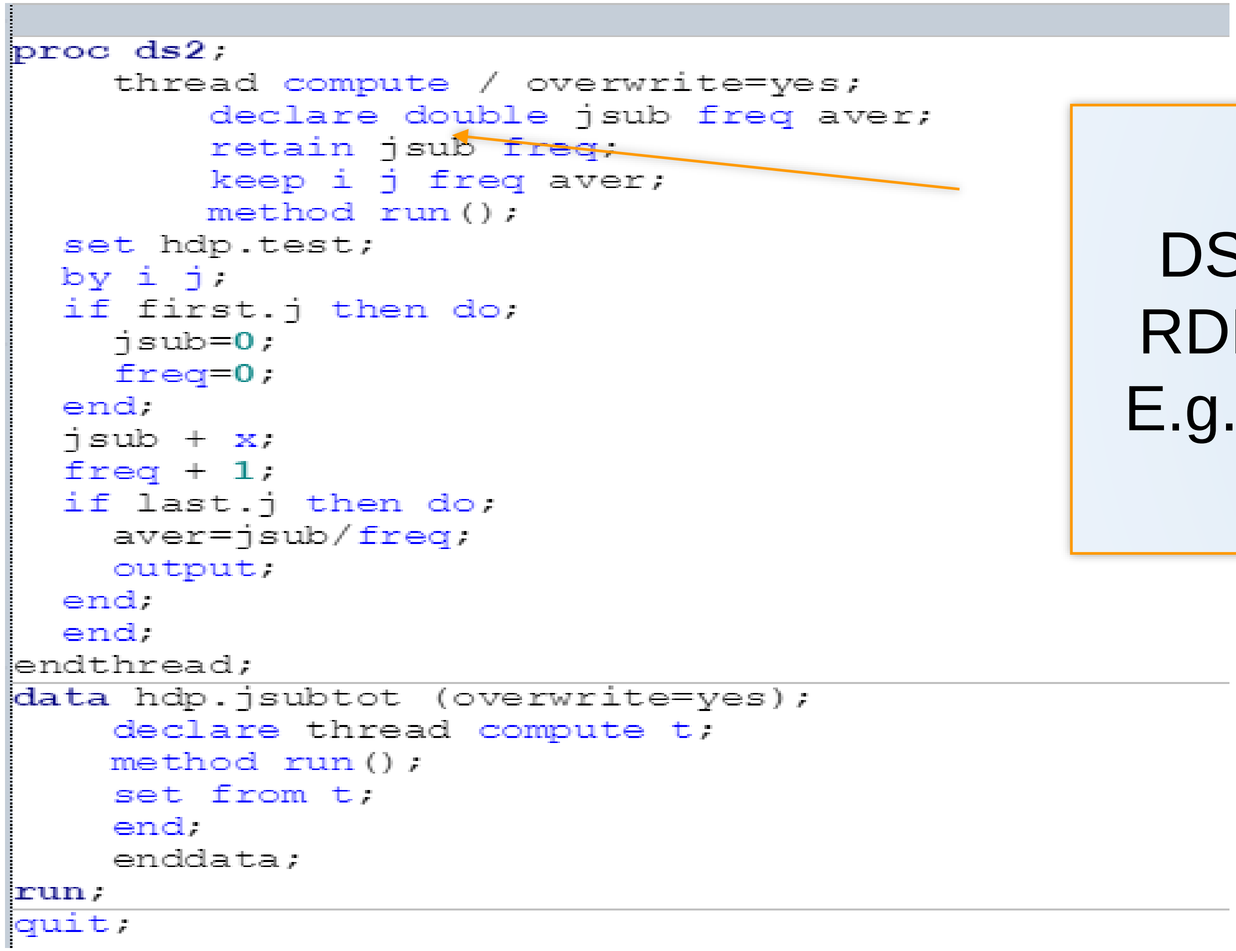
DS2 EQUIVALENT FOR HADOOP



```
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.

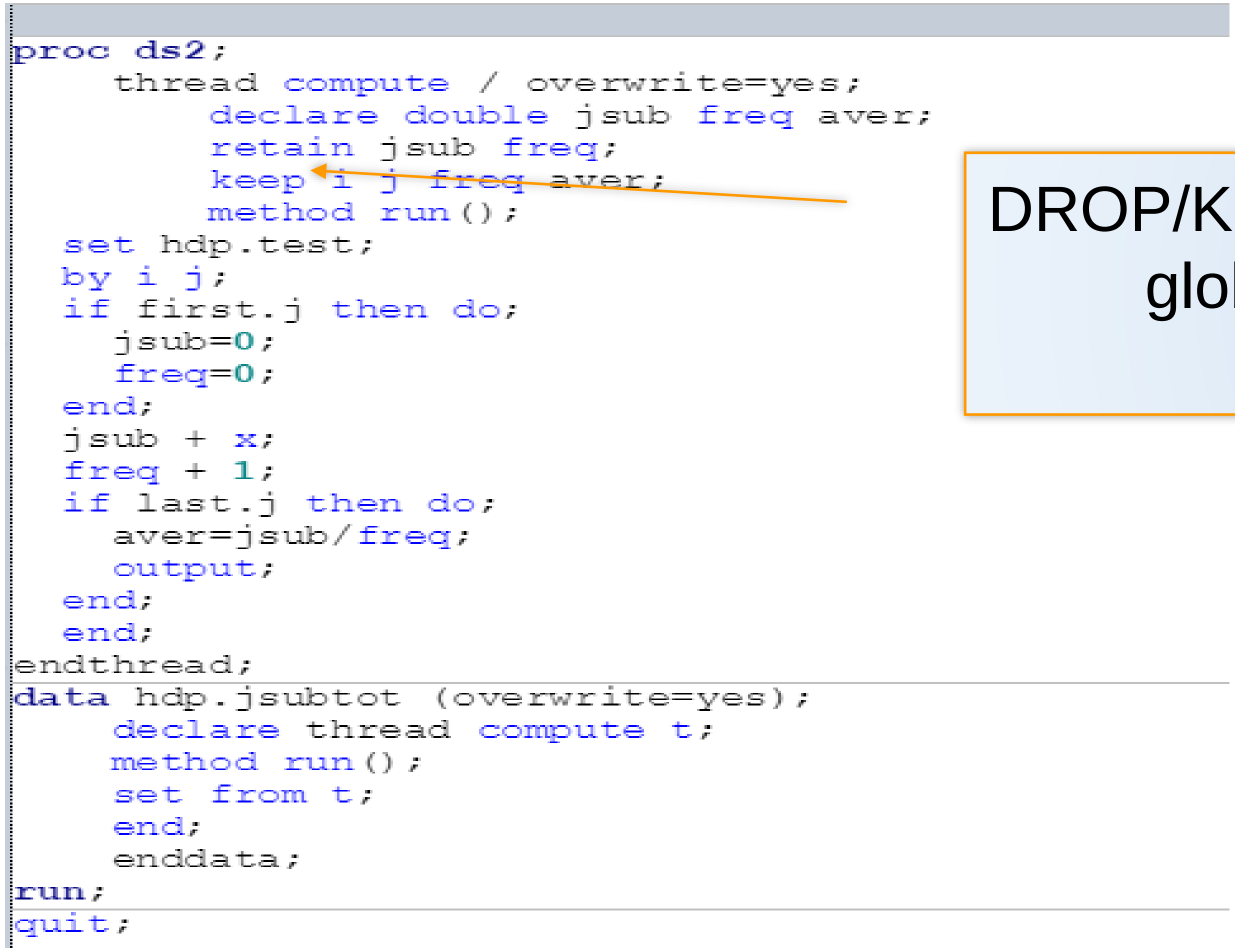
DS2 EQUIVALENT FOR HADOOP



```
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.

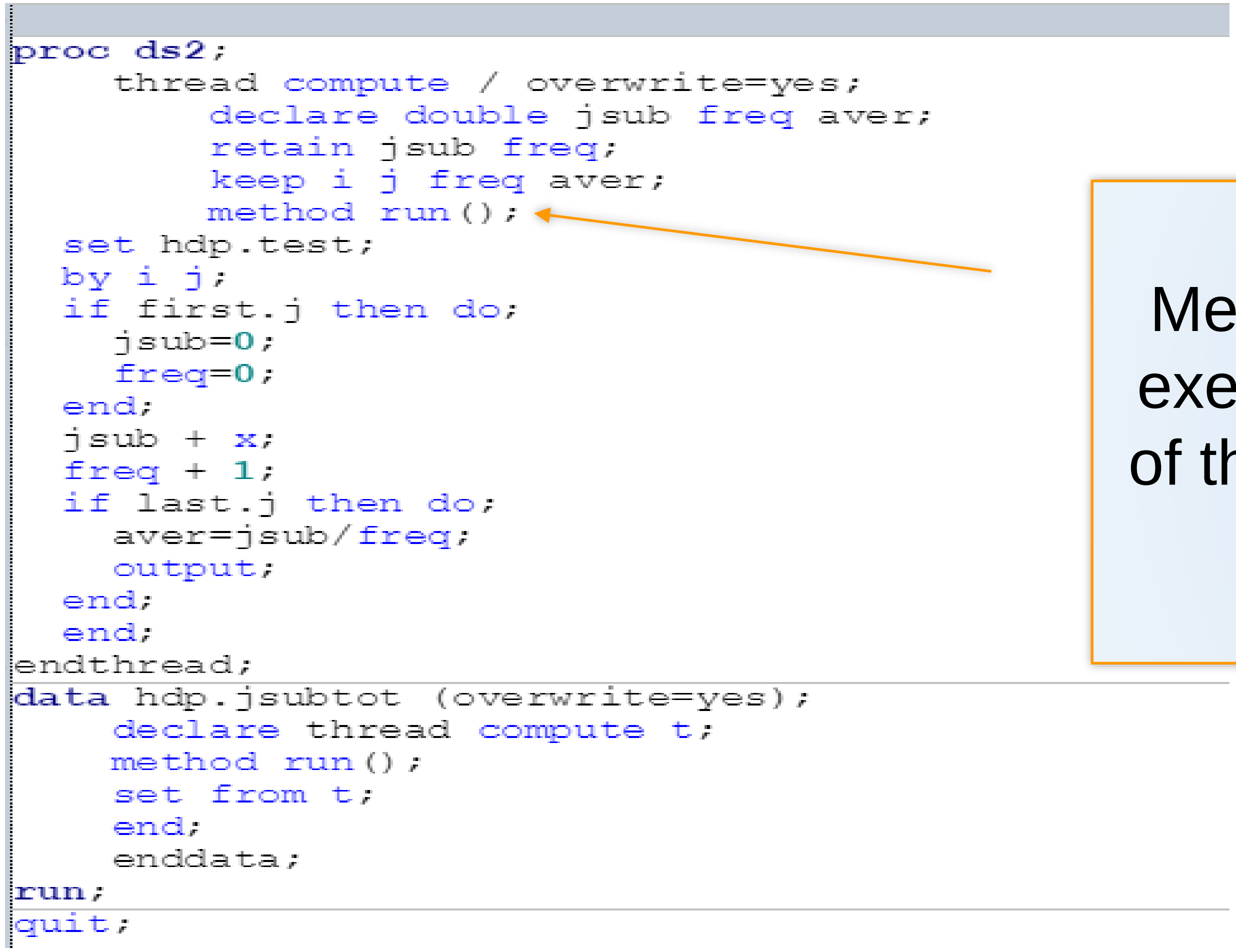
DS2 EQUIVALENT FOR HADOOP



```
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.

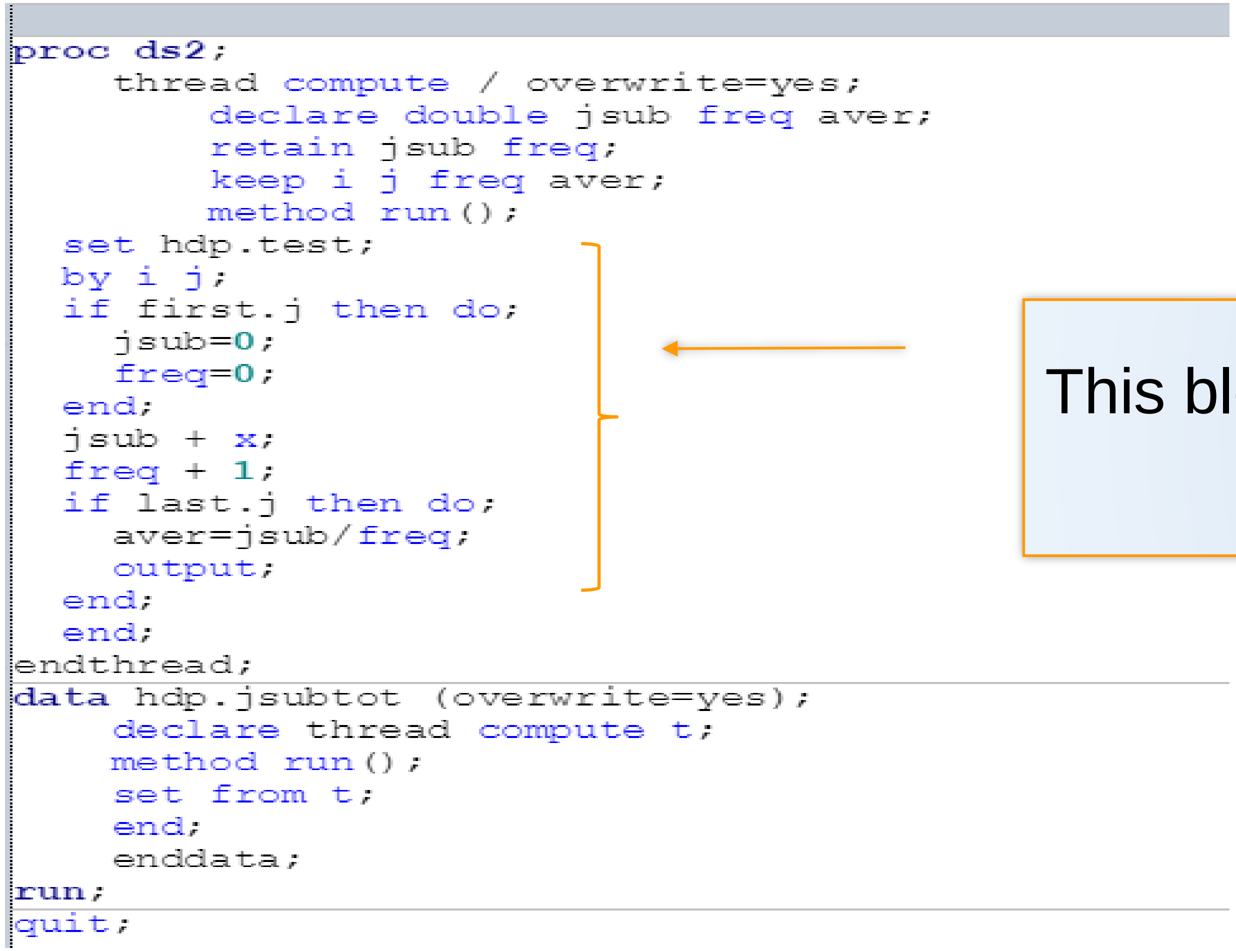
DS2 EQUIVALENT FOR HADOOP



```
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()

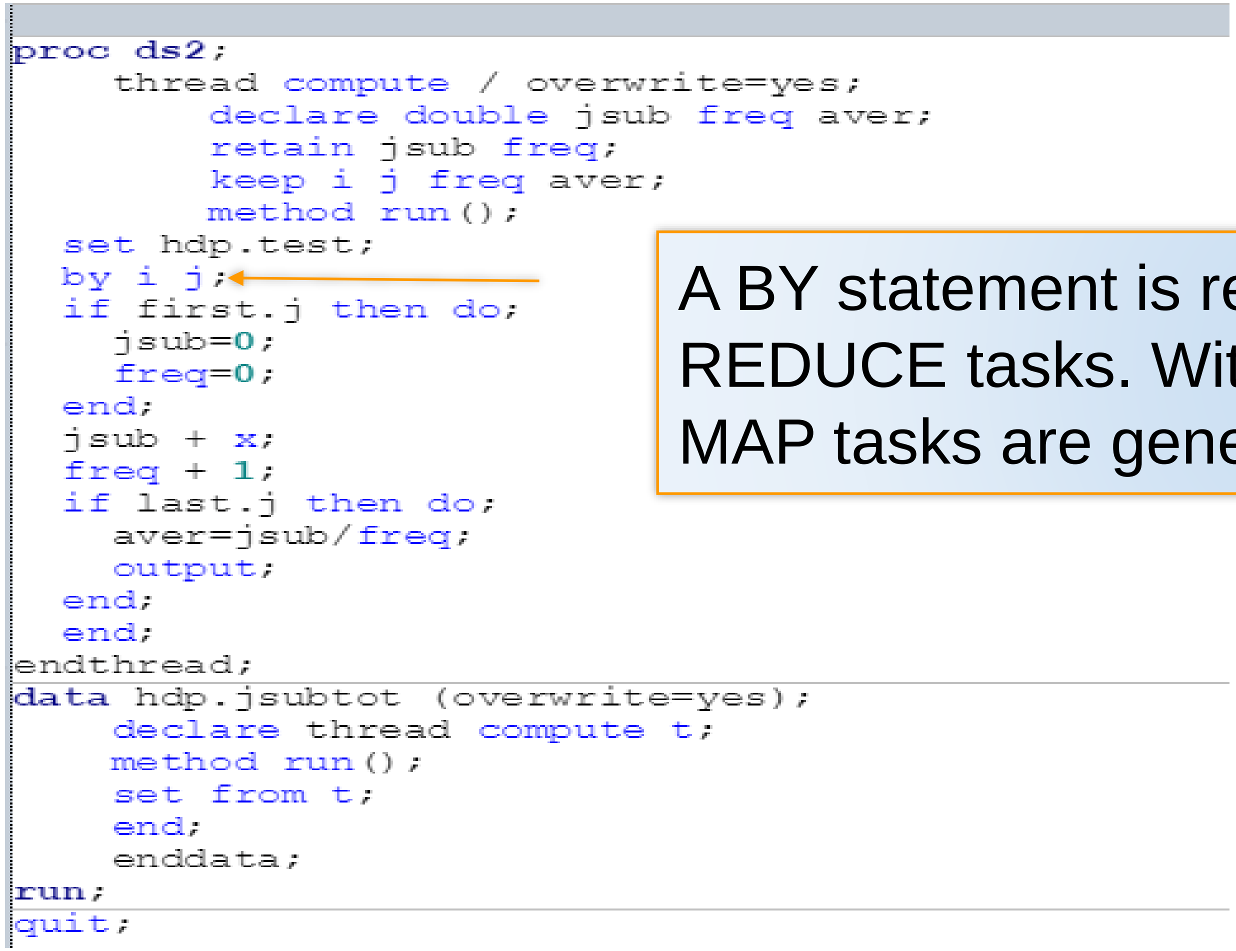
DS2 EQUIVALENT FOR HADOOP



```
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.

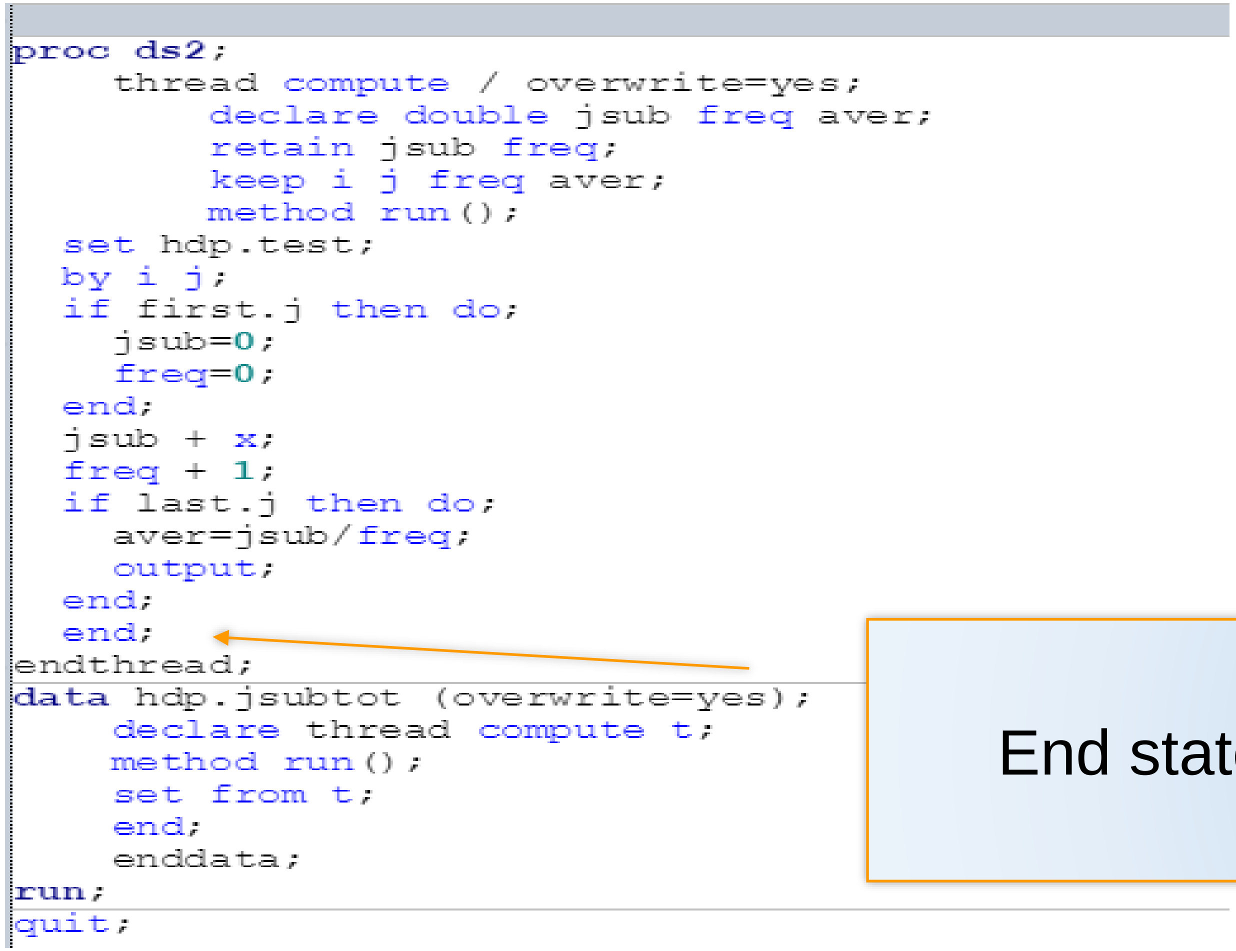
DS2 EQUIVALENT FOR HADOOP



```
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.

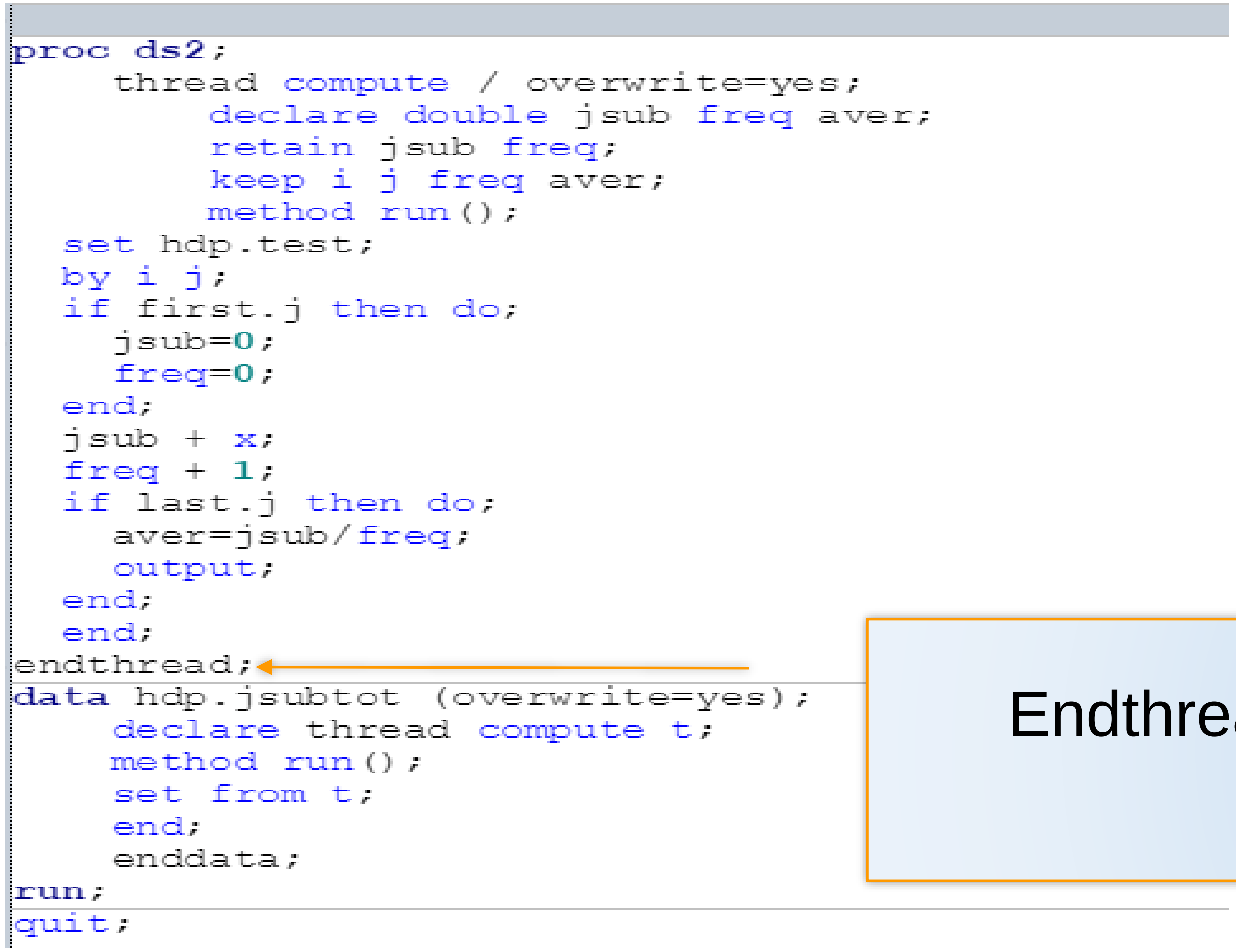
DS2 EQUIVALENT FOR HADOOP



```
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.

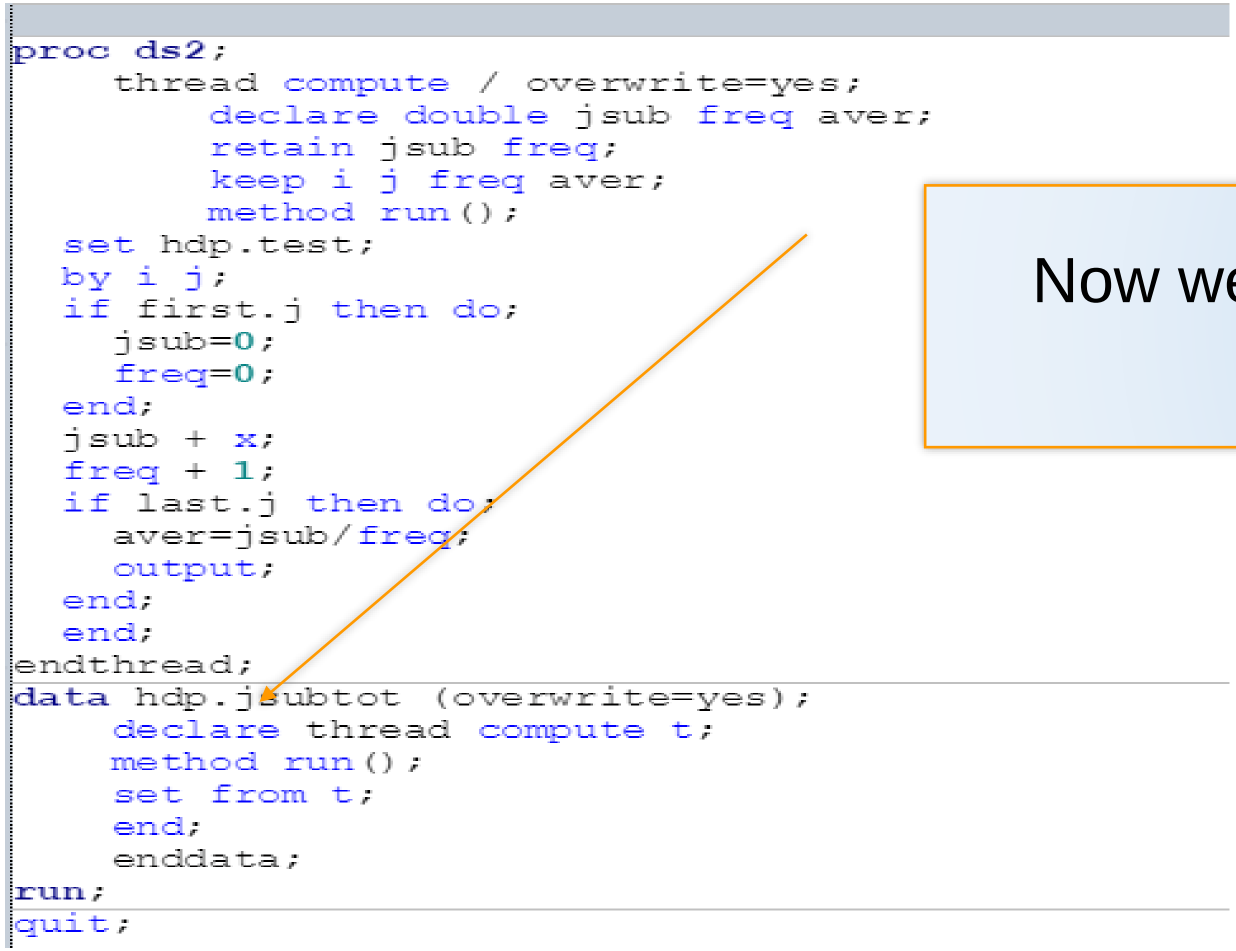
DS2 EQUIVALENT FOR HADOOP



```
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.

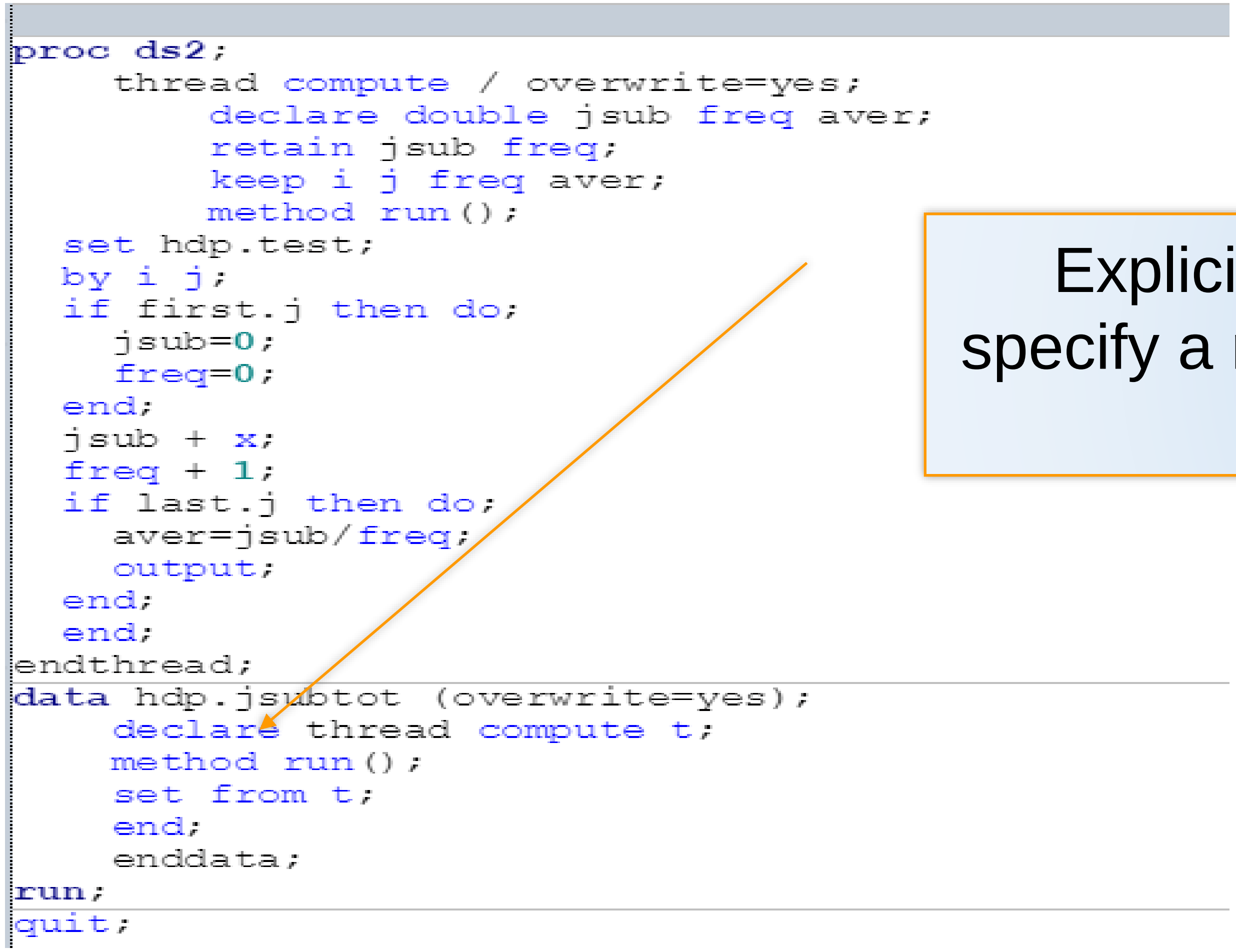
DS2 EQUIVALENT FOR HADOOP



```
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

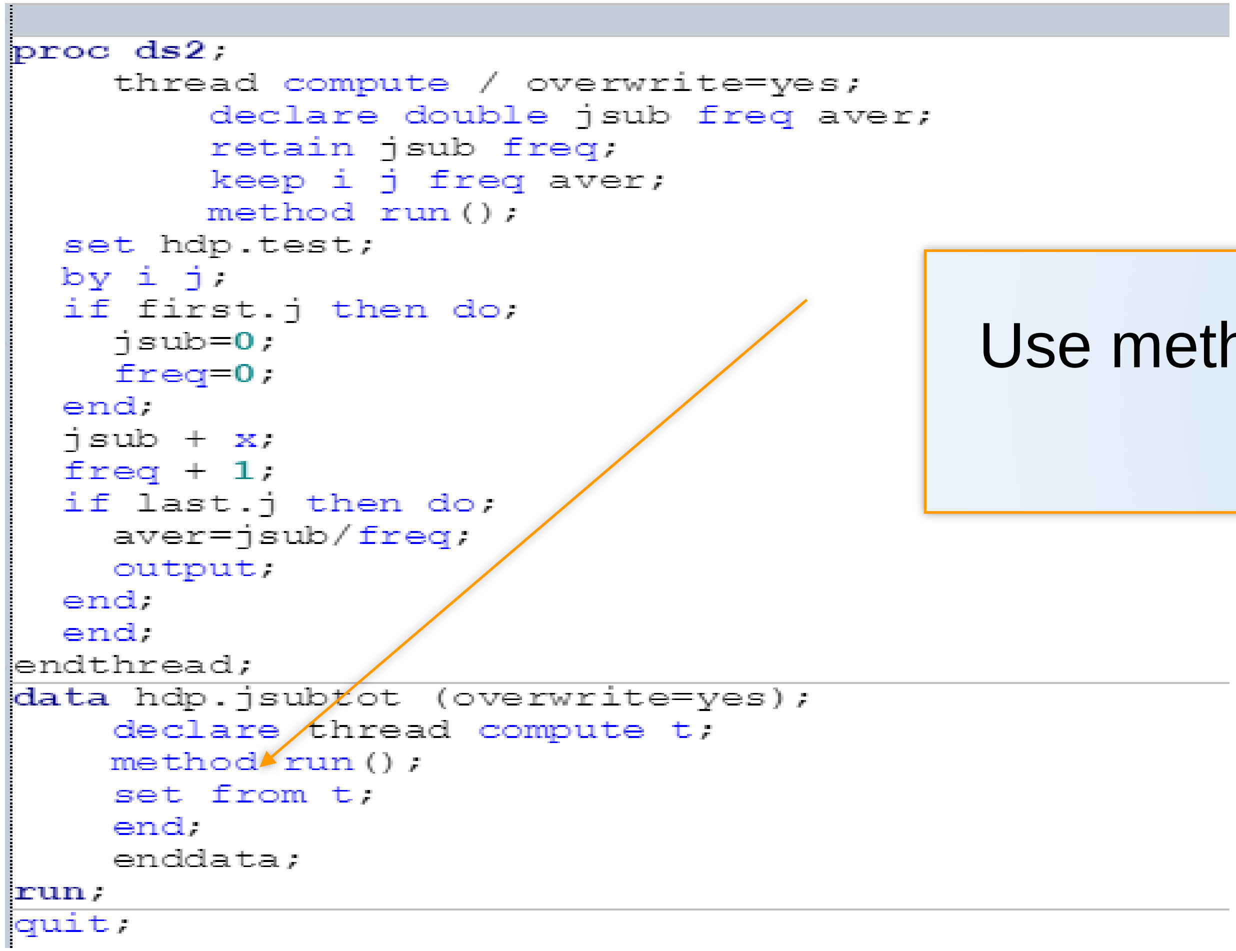
DS2 EQUIVALENT FOR HADOOP



```
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.

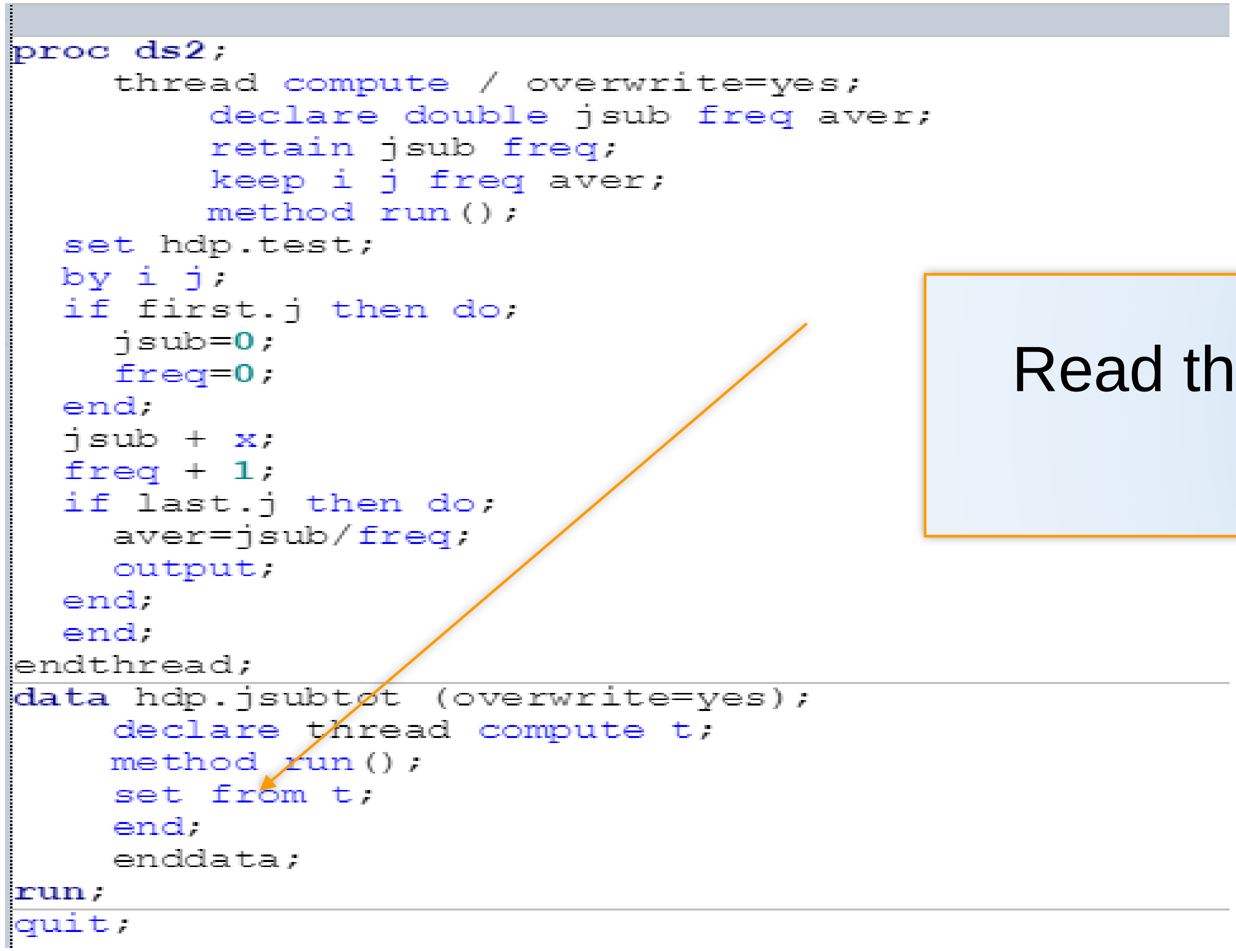
DS2 EQUIVALENT FOR HADOOP



```
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

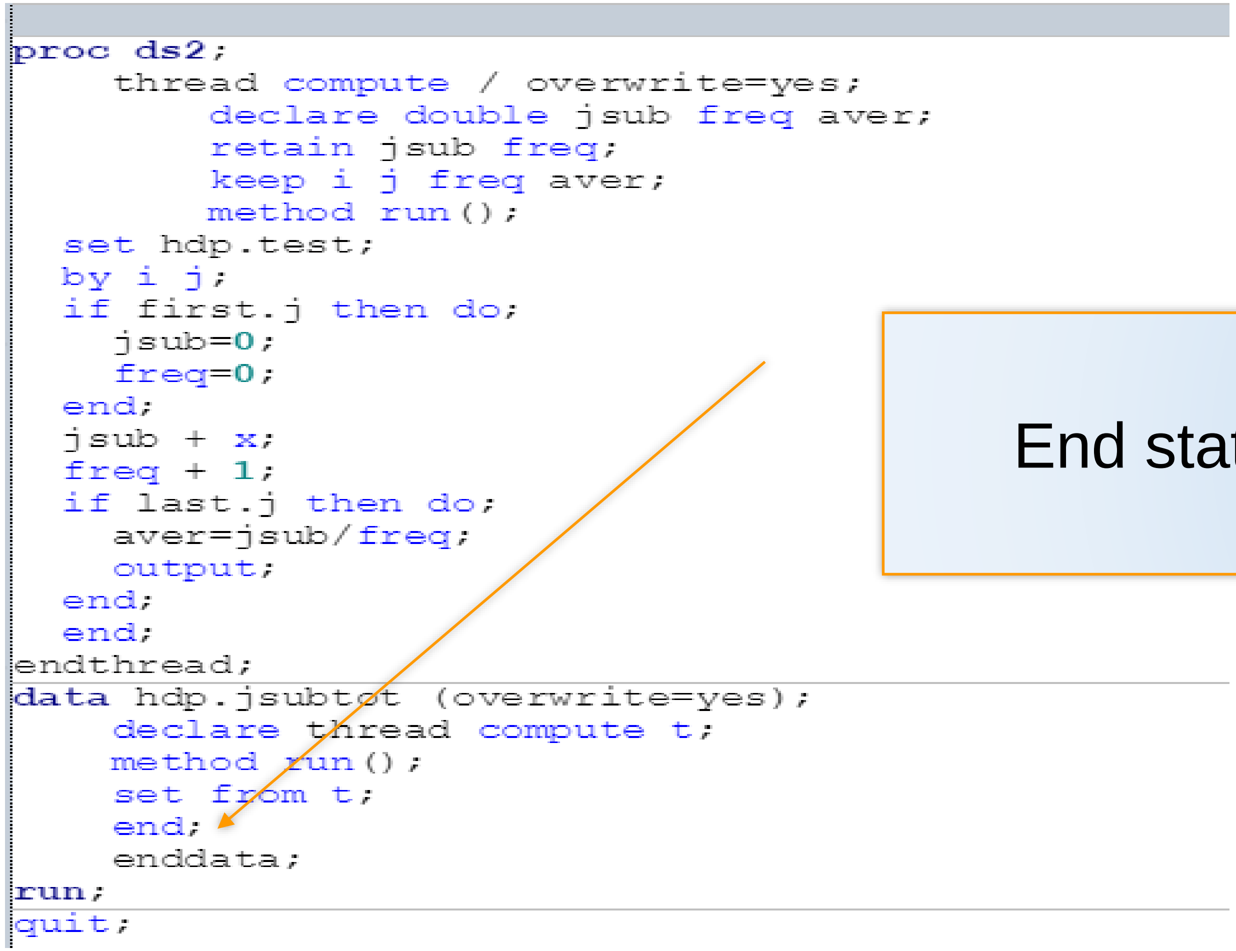
DS2 EQUIVALENT FOR HADOOP



```
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

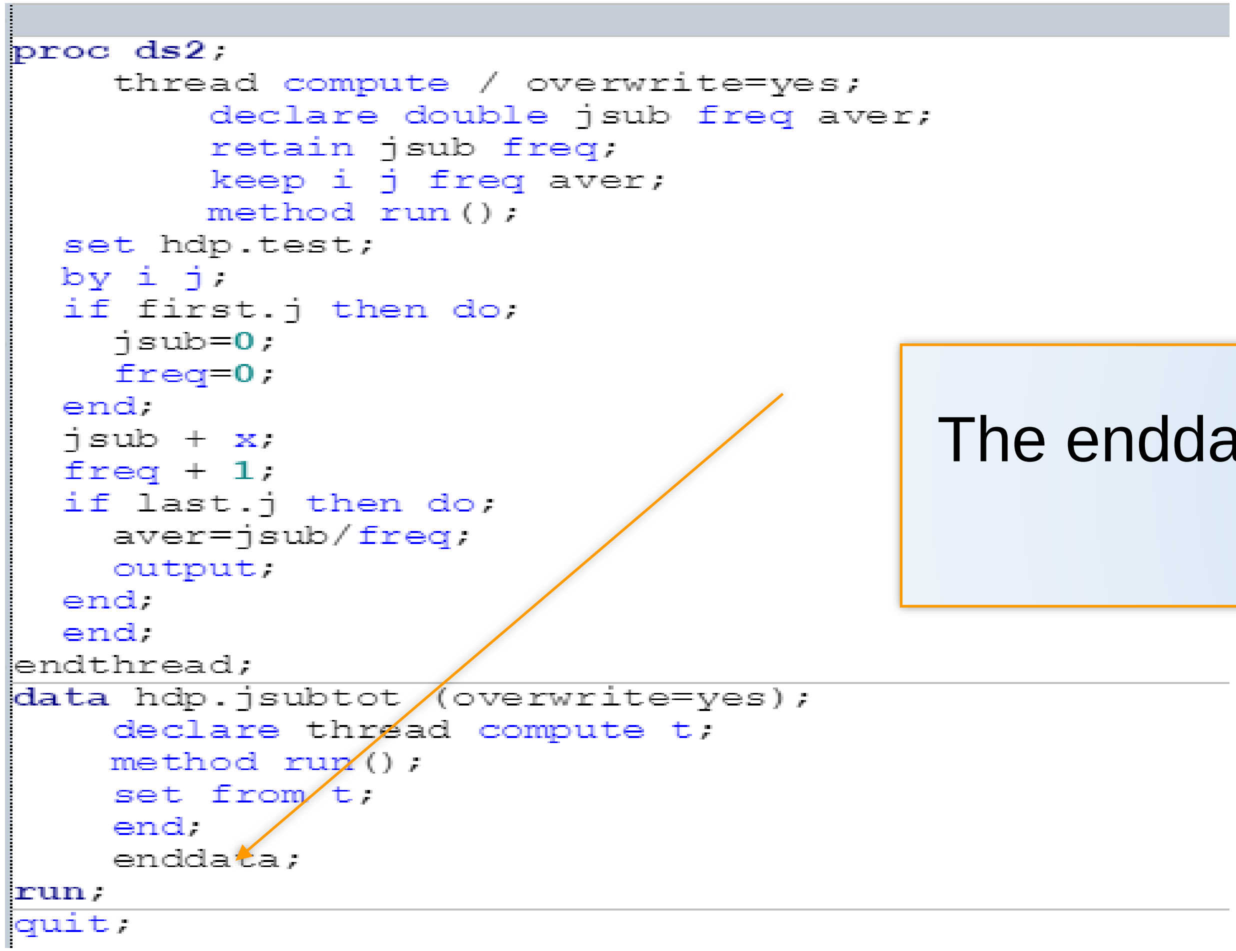
DS2 EQUIVALENT FOR HADOOP



```
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.

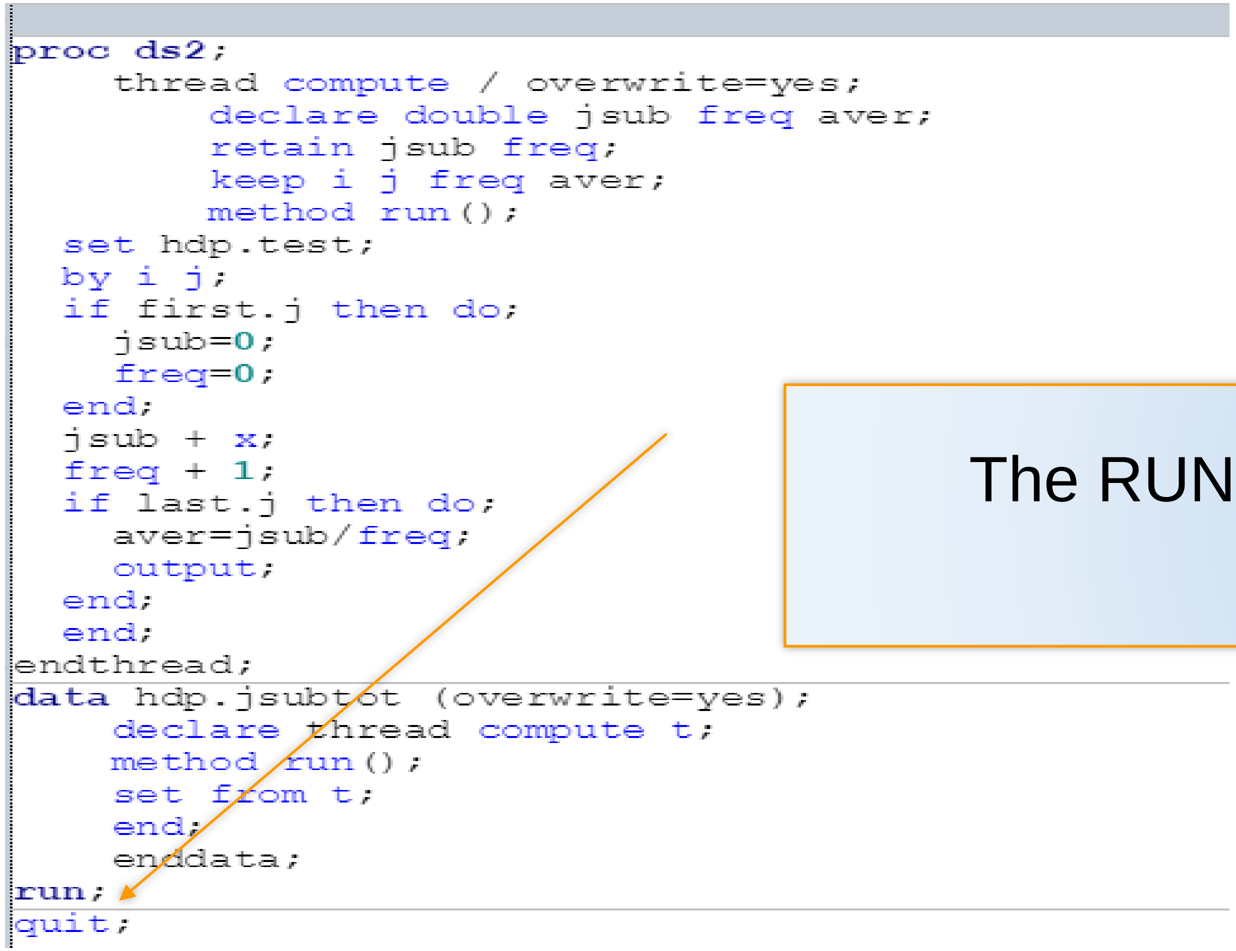
DS2 EQUIVALENT FOR HADOOP



```
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

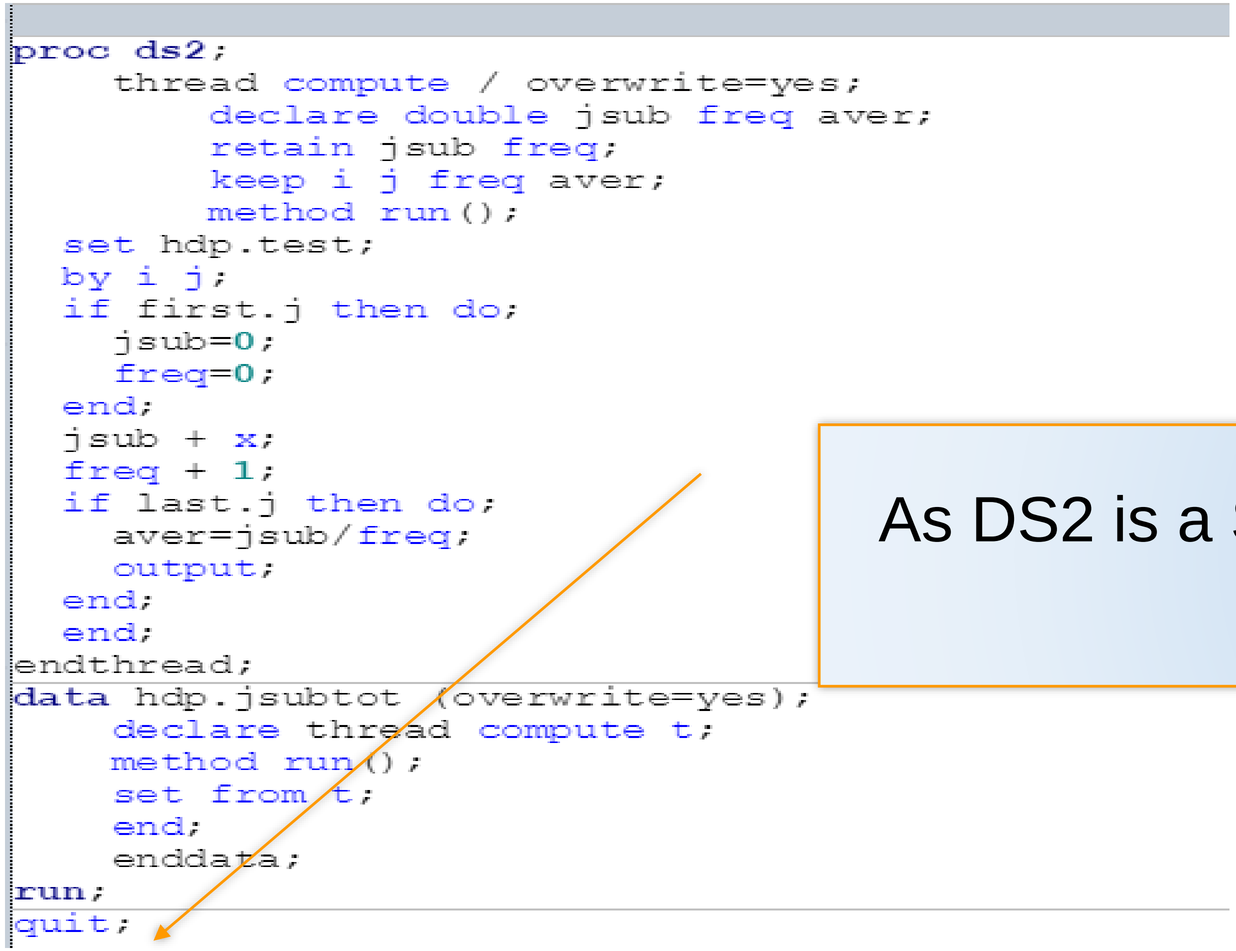
DS2 EQUIVALENT FOR HADOOP



```
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

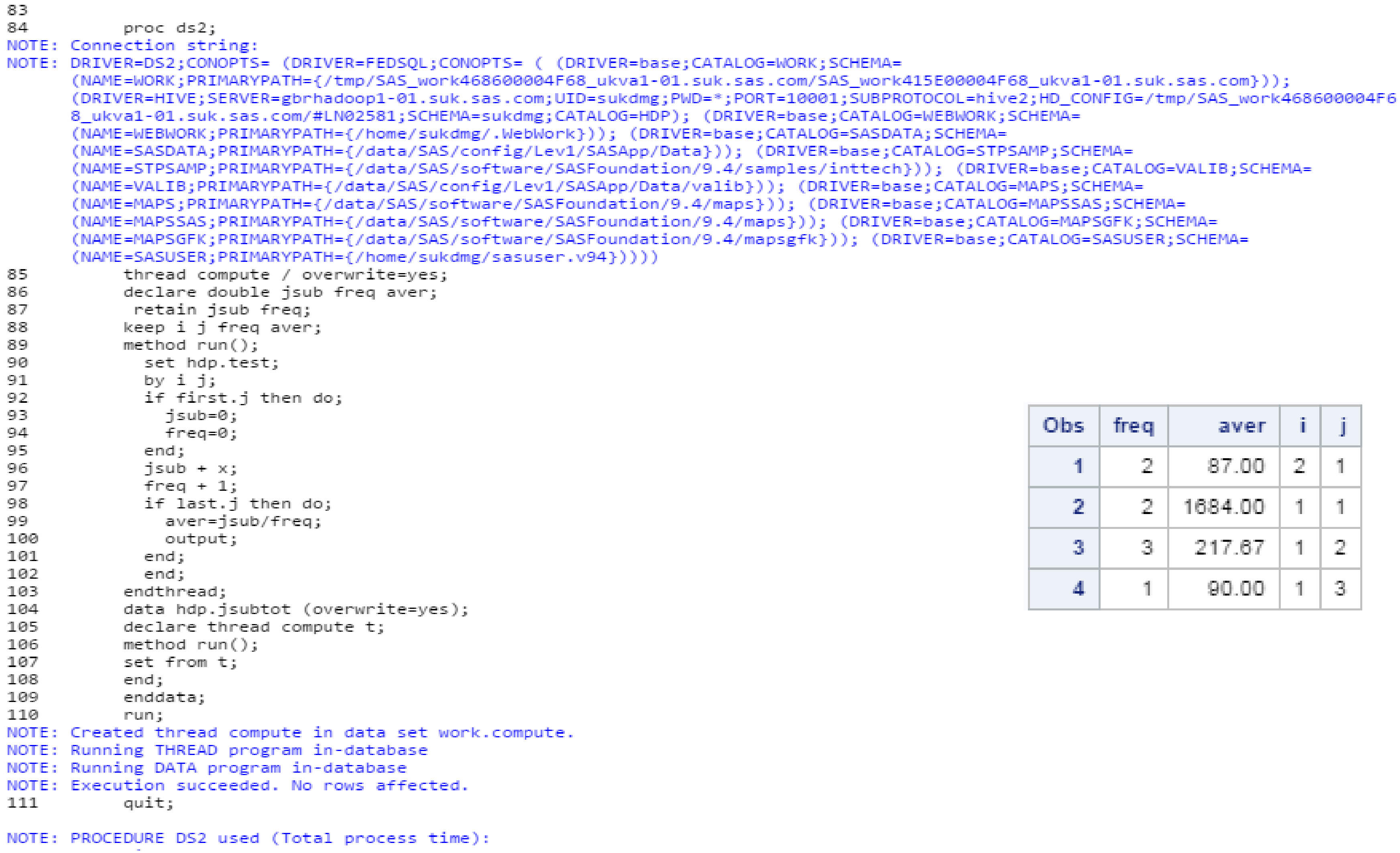
DS2 EQUIVALENT FOR HADOOP



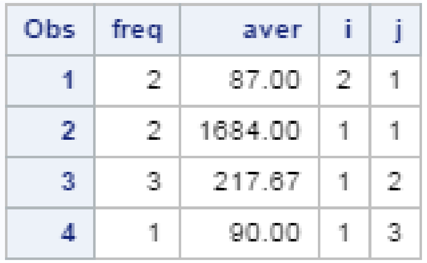
```
proc ds2;
    thread compute / overwrite=yes;
        declare double jsub freq aver;
        retain jsub freq;
        keep i j freq aver;
        method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;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

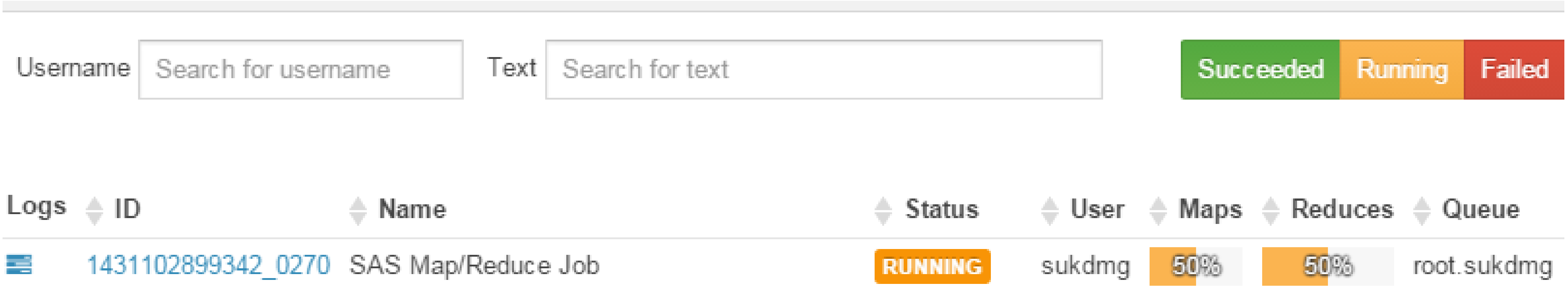


```
83
84          proc ds2;
NOTE: Connection string:
NOTE: DRIVER=DS2;CONOPTS= (DRIVER=FEDSQL;CONOPTS= ( (DRIVER=base;CATALOG=WORK;SCHEMA=
      (NAME=WORK;PRIMARYPATH={/tmp/SAS_work468600004F68_ukva1-01.suk.sas.com/SAS_work415E00004F68_ukva1-01.suk.sas.com}));
      (DRIVER=HIVE;SERVER=gbrhadoop1-01.suk.sas.com;UID=sukdmg;PWD=*;PORT=10001;SUBPROTOCOL=hive2;HD_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?



Username Search for username Text Search for text

Succeeded Running Failed

Logs	ID	Name	Status	User	Maps	Reduces	Queue
☰	1431102899342_0270	SAS Map/Reduce Job	RUNNING	sukdmg	50%	50%	root.sukdmg