SAS® GLOBAL FORUM 2015

The Journey Is Yours

An Insider's Guide to SAS/ACCESS to Hadoop

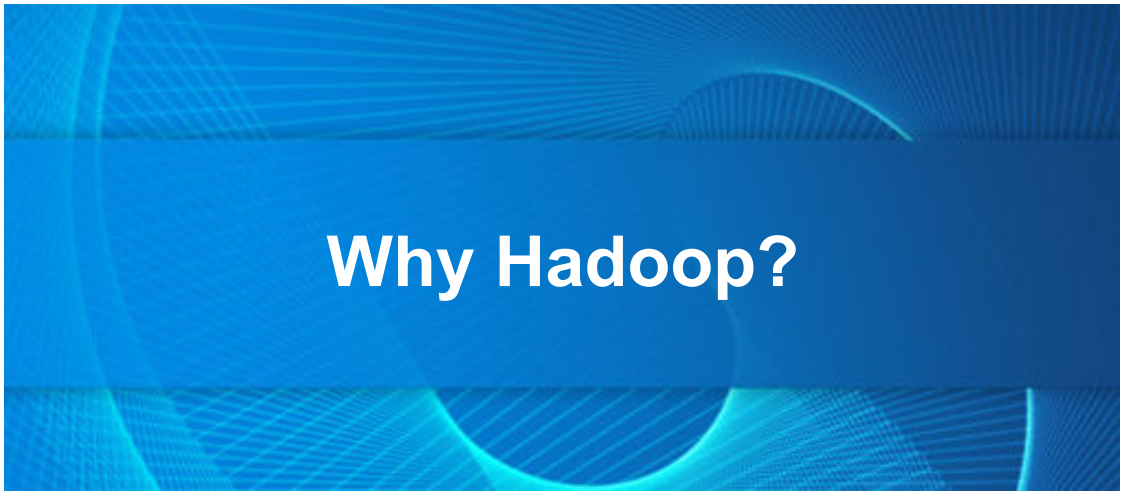
Jeff Bailey, Cloudera
Diane Hatcher, SAS
Lisa Dodson, SAS

SAS GLOBAL FORUM

WE WILL ANSWER THESE QUESTIONS

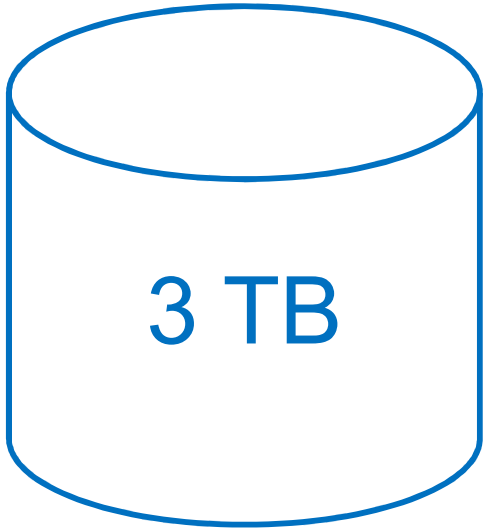
- What is Hadoop?
- How do you move files into, and out of, HDFS?
- How do you execute MapReduce code from SAS?
- How do you configure SAS/ACCESS to Hadoop?
- How do you analyze data stored in Hadoop using SAS?

SAS GLOBAL FORUM



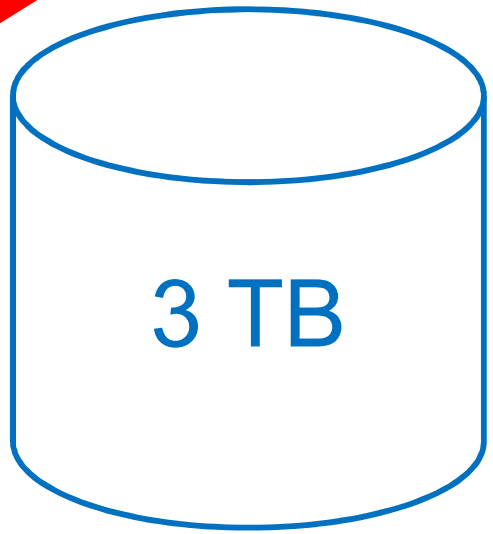
cloudera

HOW MUCH DOES THIS DRIVE COST?



HOW MUCH DOES THIS DRIVE COST?

Silly, you couldn't get a 3TB drive in 1980!

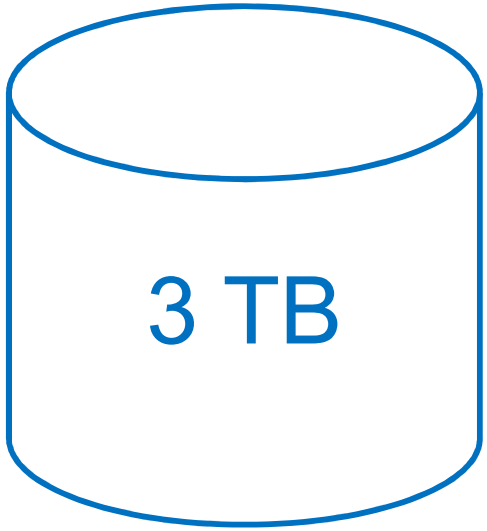


1980

$1,312,500,000

HOW MUCH DOES THIS DRIVE COST?

That's $0.03 per GB!



TODAY	$92
2010	$270
2005	$3,720
2000	$33,000
1995	$3,360,000
1990	$33,600,000
1985	$315,000,000
1980	$1,312,500,000

SAS GLOBAL FORUM

HOW MUCH DOES THIS DRIVE COST?

That's $0.03 per GB!

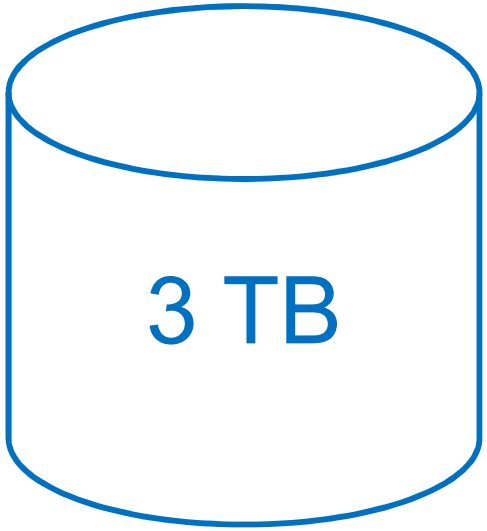
TODAY	$92
2010	$270
2005	$3,720
1980	$1,312,500,000

Insight: Disk Space is FREE!

SAS GLOBAL FORUM

IT'S NOT JUST ABOUT COST!

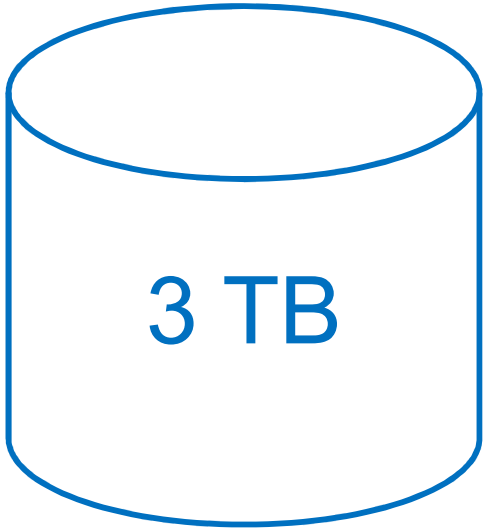
How long does it take to read **3 TB of data**?



SAS GLOBAL FORUM

IT'S NOT JUST ABOUT COST!

How long does it take to read **3 TB of data**?



4.17 Hours

SAS GLOBAL FORUM

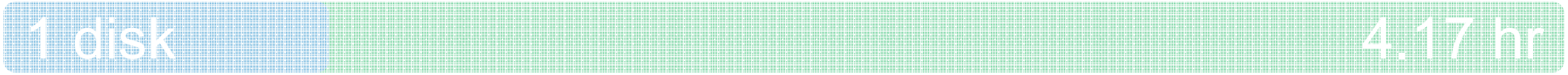
IT'S NOT JUST ABOUT COST!

How long does it

What happens if you add more disks?

HOW LONG DOES IT TAKE TO READ A 3TB FILE?

HOW LONG DOES IT TAKE TO READ A 3TB FILE?



Insight: More Disks are FASTER!

SAS GLOBAL FORUM

What is Hadoop?

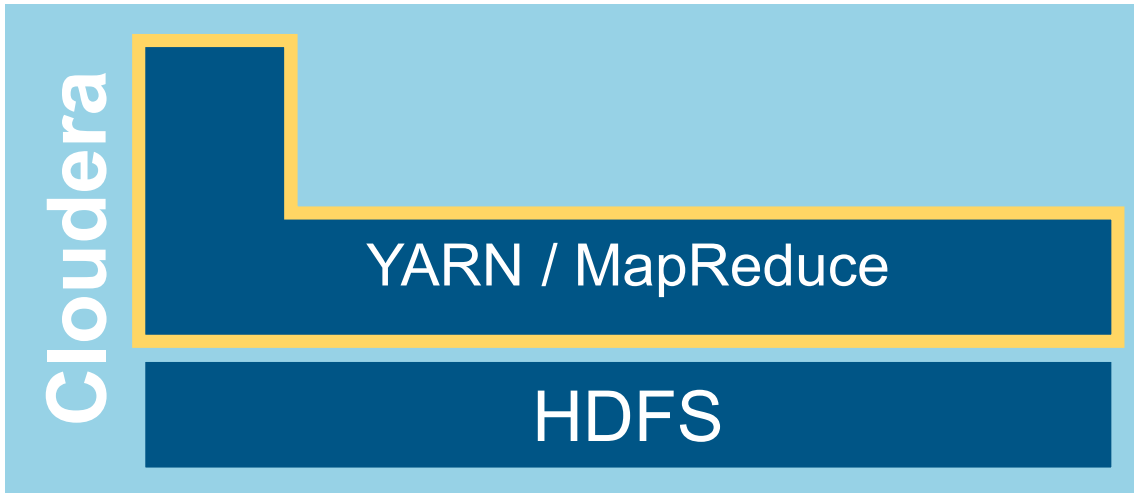
cloudera®

HADOOP IS A STORAGE PLATFORM

- Distributed Storage Performs Great
- Data is Replicated
- Reasonable Cost
- Sits on the OS File System

SAS GLOBAL FORUM

HADOOP IS A PROCESSING PLATFORM



- MapReduce/YARN
- Distributed Processing
- Data Locality
- Usually Java

SAS GLOBAL FORUM

HADOOP IS A PROCESSING PLATFORM

MapReduce has 3 phases:

Map Phase

- Reads Data
- Key/Value Pairs

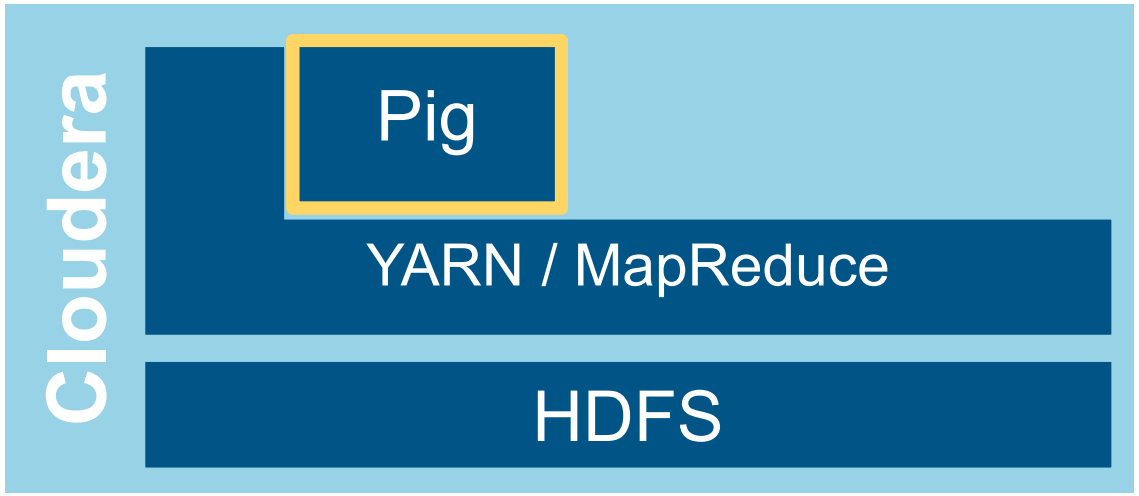
Shuffle + Sort

- Moves data between nodes to prepare for the reduce phase

Reduce Phase

- Combines intermediate keys
- Summary/Aggregation

Apache Pig

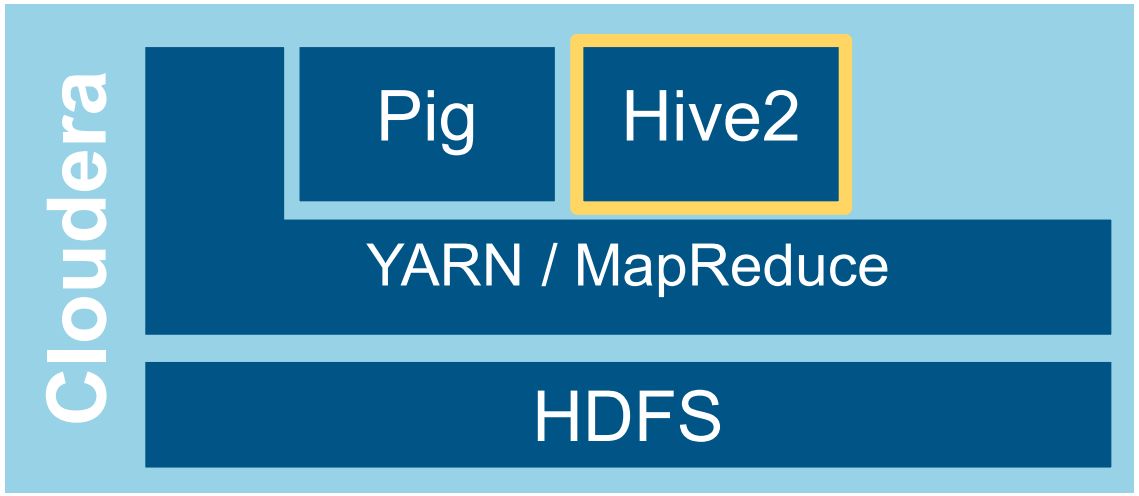


- Scripting Language
- Higher level than programming Java MapReduce
- Pig Latin scripts are converted to MapReduce jobs
- Great for joining data
- Great for transforming data

Apache Pig: Example Program

```
people = LOAD '/user/training/customers' AS (cust_id, name);
orders = LOAD '/user/training/orders' AS (ord_id, cust_id, cost);
groups = GROUP orders BY cust_id;
totals = FOREACH groups GENERATE group, SUM(orders.cost) AS t;
result = JOIN totals BY group, people BY cust_id;
DUMP result;
```

Apache Hive



- SQL on Hadoop
- Similar to traditional SQL
- Reduces development time
- Enables BI on Hadoop
- Schema-on-Read
- You choose underlying file format

Apache Hive

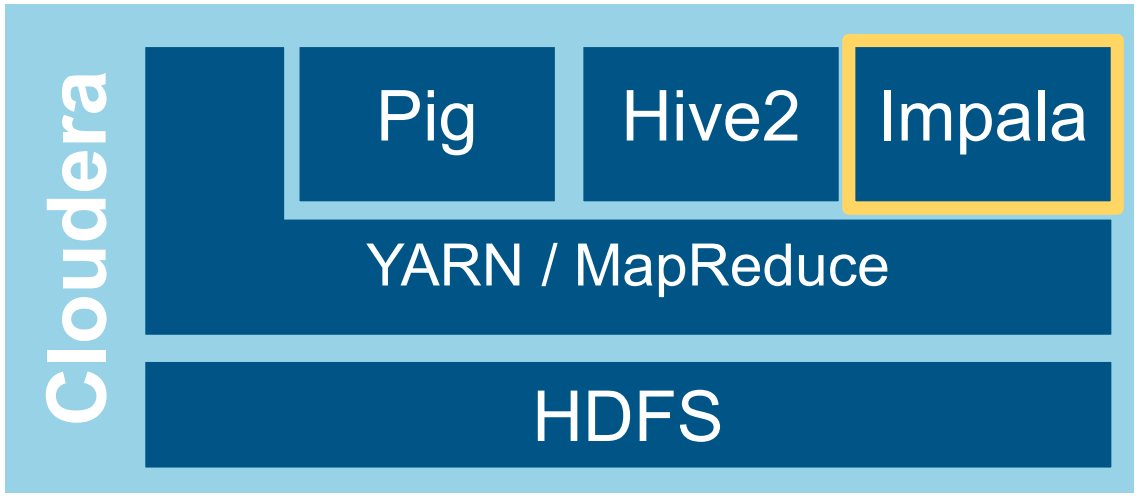
- SQL on Hadoop
- Similar to traditional

```
SELECT zipcode, SUM(cost) AS total
FROM customers
JOIN orders
ON (customers.cust_id = orders.cust_id)
WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```

Comparing Hive to an RDBMS

	Hive	Relational Database
Query language	SQL (subset)	SQL (full)
Update individual records	No	Yes
Delete individual records	No	Yes
Append records	Yes	Yes
Transactions	No	Yes
Index support	Limited	Extensive
Latency	High	Very low
Data size	Petabytes	Terabytes

Impala



- High-performance SQL engine
- Handles concurrency well
- Does not rely on MapReduce
- Supports a dialect of SQL very similar to Hive's
- 100% open source
- Apache License

Comparing Impala to Hive and RDBMS

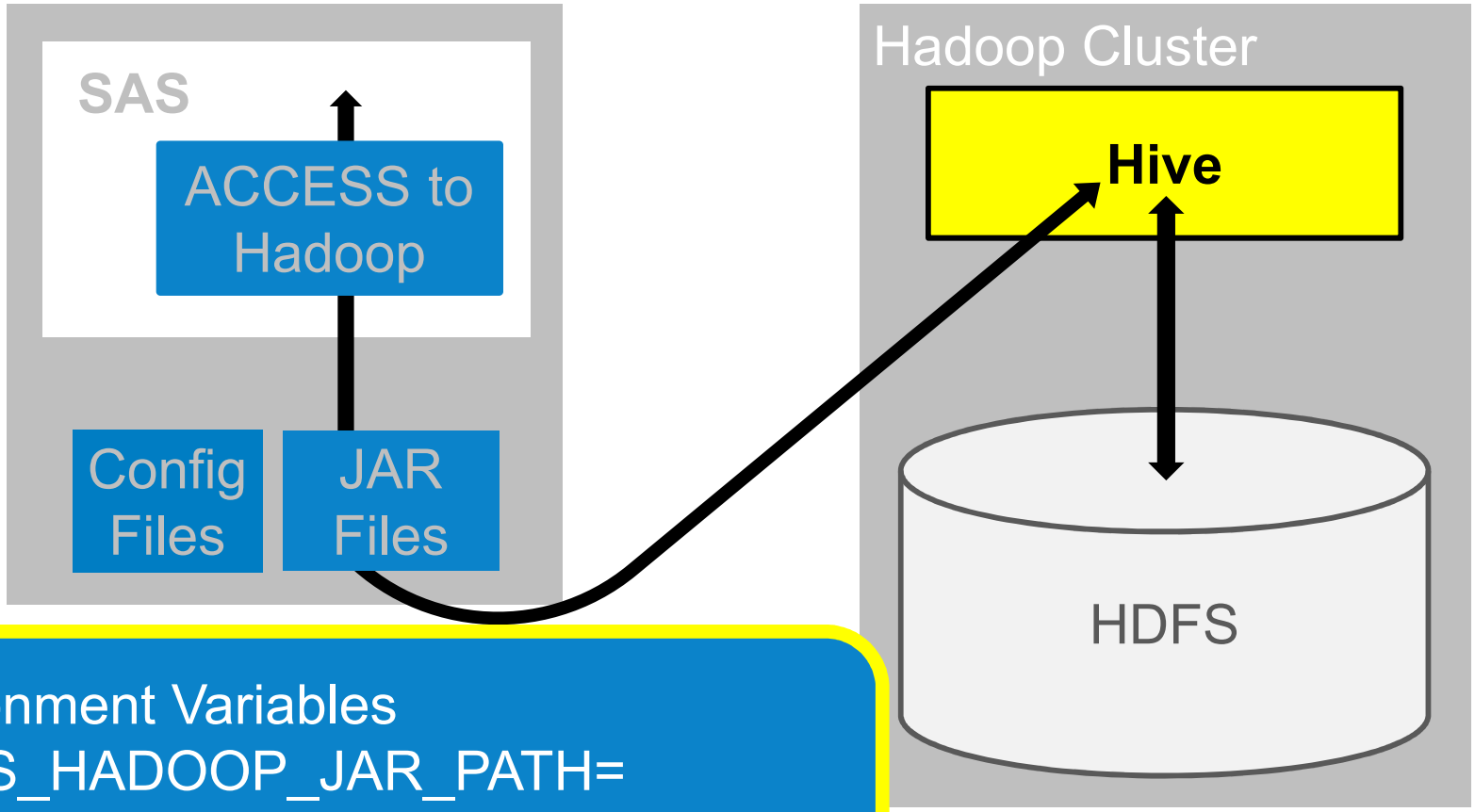
	Impala	Hive	Relational Database
Query language	SQL (subset)	SQL (subset)	SQL (full)
Update individual records	No	No	Yes
Delete individual records	No	No	Yes
Append records	Yes	Yes	Yes
Transactions	No	No	Yes
Index support	No	Limited	Extensive
Latency	Low	High	Very low
Data size	Petabytes	Petabytes	Terabytes

SAS/ACCESS to Hadoop Configuration Overview

cloudera

sas

HOW DOES SAS CONNECT TO HADOOP?



Environment Variables
- SAS_HADOOP_JAR_PATH=
- SAS_HADOOP_CONFIG_PATH=
- SAS_HADOOP_RESTFUL=

Optional – used with webHDFS

WHEN SHOULD YOU USE A CONFIG FILE?

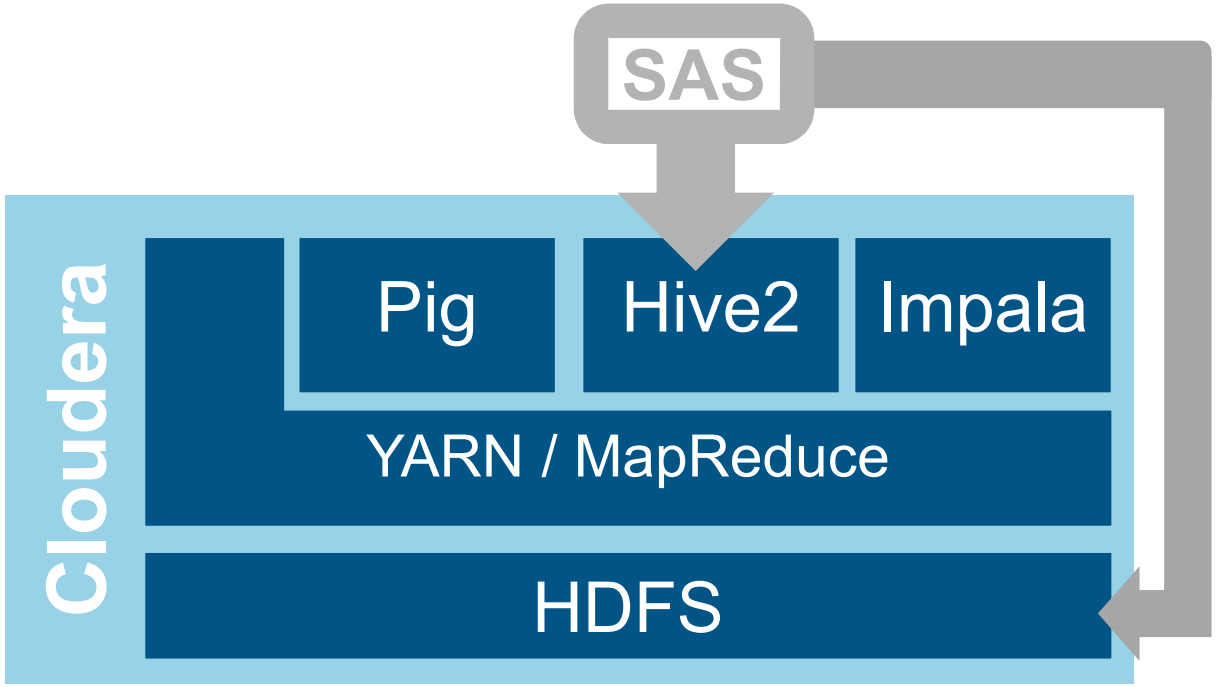
CFG= Is Required When …

- You are running SAS 9.4m1 or earlier.
- HDFS + Hive are running on different machines
- HDFS failover is running
- Non-HDFS based Hadoop environment
- You need to change an Hadoop property
- You need to specify the version of MapReduce

CFG= option has been deprecated with 9.4m2 (which uses the environment variable options)

SAS GLOBAL FORUM

SAS/ACCESS TO HADOOP



- Connects via JDBC
- Makes Hive tables look like SAS data sets
- Converts SAS code to HiveQL
- Can use webHDFS
- Bulk loads directly to HDFS

SAS GLOBAL FORUM

Configuration Exploration

This demonstration illustrates how to configure SAS/ACCESS to Hadoop.

Reference:
http://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoopbacg.pdf

SAS/ACCESS to Hadoop
LIBNAME Statement

cloudera

sas

LIBNAME STATEMENT

```
/* SAS 9.4M2 and Later: SUBPROTOCOL default is hive2 */
/* SAS 9.4M1 and earlier: SUBPROTOCOL default is hive */

LIBNAME mycdh HADOOP SERVER=cdhserver
                     user=cloudera
                     PORT=10000
                     SUBPROTOCOL=hive2;
```

Passing Hadoop Configuration Parameters

```
LIBNAME mycdh HADOOP SERVER=cdhserver
    PROPERTIES="mapreduce.map.memory.mb=2048";
```

Exercises

- Environment includes the following:
  - Cloudera 5.3 Quickstart VM, running in vmware VM Player 7 (free edition).
  - SAS 9.4m2
    - » SAS Studio – Basic *(can also use Single User version, SAS Display Manager, or SAS Enterprise Guide)*
    - » SAS/ACCESS to Hadoop *(configured to point to Hadoop VM)*
  - Configured desktop “hosts” file to point to Quickstart VM’s IP address as “quickstart.cloudera”
  - SAS Data Loader Free Trial version *(separate exercises)*
    - » *SAS Data Loader is available from http://www.sas.com/dataloader*

Exercise 1 – Inventory Hadoop

This exercise reinforces the concepts discussed previously.

Exercise 1

- Create libname statement
  - Open a program window in your SAS application.

```
/*
/* Define libname for SAS HOW class
*/
libname mycdh hadoop server='quickstart.cloudera' schema='default' user=cloudera;
```

- Libref = mycdh
- Engine = hadoop
- Server = quickstart.cloudera (defined in hosts file)
- Specifying schema is optional. This environment has multiple schemas, so it's stated here to point to right one.
- User = cloudera (default user in Cloudera Quickstart VM)

SAS GLOBAL FORUM

Exercise 1

- Check inventory of tables in your library
  - Add the following code

```
/*
/* proc datasets = listing of tables
/* proc contents = table details
*/
proc datasets lib=mycdh details;
run;

proc contents data=mycdh._all_;
run;
```

SAS GLOBAL FORUM

Implicit Pass-Through

How Does SAS/ACCESS Talk to Hadoop?

```
proc sql;
   select count(*) from cars
      where make='BMW';
run;
```

?

SAS GLOBAL FORUM

SAS/ACCESS Uses SQL to Talk to Hive2

```
proc sql;
   select count(*) from mycdh.cars
      where make='BMW';
run;
```

```
select COUNT(*) from 'sasschema'.`CARS` TXT_1
WHERE TXT_1.'make' = 'BMW'
```

SAS Generated This SQL

Use the SAS OPTIONS Statement to See SQL

```
proc sql;
   select count(*) from mycdh.cars;
      where make='BMW';
run;
```

```
OPTIONS  SASTRACE=',,,d' SASTRACELOC=SASLOG NOSTSUFFIX;
```

```
select COUNT(*) from 'sasschema'.`CARS` TXT_1
WHERE TXT_1.`make` = 'BMW'
```

SAS Generated This SQL

Understanding SASTRACE= Output

```
HADOOP_53: Prepared: on connection 2
SHOW TABLES 'CARS'

HADOOP_54: Prepared: on connection 2
DESCRIBE FORMATTED CARS

HADOOP_55: Prepared: on connection 2
SELECT * FROM `CARS`

HADOOP_56: Executed: on connection 3
USE `sasschema`

HADOOP_57: Prepared: on connection 3
 select COUNT(*) from `CARS` TXT_1
where TXT_1.`make` = 'BMW'

HADOOP_58: Executed: on connection 3
 select COUNT(*) from `CARS` TXT_1 where
TXT_1.`make` = 'BMW'
```

Does the table exist?

Get the extended table attributes

Get the column information

Execute the SQL code

Does nothing. Listed for consistency with other engines

Exercise 2 – SASTRACE

This exercise reinforces the concepts discussed previously.

Exercise 2

- Create libname statement
  - Add the SASTRACE option to your code.

```
/*
/* Add SASTRACE option to see how code is generated and pushed down to Hadoop cluster
*/
options SASTRACE=',,,d' SASTRACELOC=SASLOG NOSTSUFFIX;
```

- To turn SASTRACE off: `options SASTRACE=off;`

Exercise 2

- Add sashelp.cars
  - use SAS data step, or
  - use PROC SQL, or
  - use SAS interface to drag and drop table from one library to another.

```
proc sql; drop table mycdh.cars; quit; /* This is optional */

/* SAS data step example
*/
data mycdh.cars; set sashelp.cars;
run;

/*
/* Proc SQL example
proc sql; create table mycdh.cars as select * from sashelp.cars;
quit;
*/
```

Exercise 2

- Add the following queries
  - Compare the SAS logs

```
/* Compare SASTRACE logs for these 2 queries
*/
proc sql;
          select * from mycdh.cars where make='BMW';
quit;

/* This one takes a bit longer to run and returns a single number
*/
proc sql;
          select count(*) from mycdh.cars where make='BMW';
quit;
```

Generates HiveQL

Generates MapReduce

Passing Joins to Hive

cloudera

sas

Passing Joins Using Implicit Pass-Through

- Cross schema joins are NOT supported
- SCHEMA= must be the same for multi-LIBNAME joins

Passing Joins Using Implicit Pass-Through

- Cross schema joins are NOT supported
- SCHEMA= must be the same for multi-LIBNAME joins

```
LIBNAME mycdh1 HADOOP SERVER=cdhserver USER=myuser;
LIBNAME mycdh2 HADOOP SERVER=cdhserver USER=myuser;
```

JOIN will Pass-Thru

```
LIBNAME mycdh1 HADOOP SERVER=cdhserver USER=myuser;
LIBNAME mycdh2 HADOOP SERVER=cdhserver;
```

What About This One?

SAS GLOBAL FORUM

Was the Join Passed to Hadoop?

Check the SAS log:

ACCESS ENGINE: SQL **statement was passed** to the DBMS for fetching data.

JOIN Did Pass

ERROR: This SQL **statement will not be passed** to the DBMS for processing because it involves a join across librefs with different connection properties.

JOIN Did Not Pass

Exercise 3 - Join

This exercise reinforces the concepts discussed previously.

Exercise 3

- Open a new program window and add the following code (next 2 slides). Run.
  - Which joins work?
- Copy included SAS datasets (customers, orders) to your machine.
  - Example directory: C:\SASGF\HOWData

Exercise 3

```
/* Create some additional libname statements
*/
libname howdata 'C:\SASGF\HOWData';
libname mycdh2 hadoop server='quickstart.cloudera' schema='default';

/* Add tables "customers" and "orders", if they don't exist yet in
mycdh library
*/
proc SQL;
  create table mycdh.customers as select * from howdata.customers;
  create table mycdh.orders as select * from howdata.orders;
quit;

/* Add tables "customers", if it doesn't exist yet in mycdh3 library
*/
proc SQL;
  create table mycdh3.customers as select * from howdata.customers;
quit;
```

SAS GLOBAL FORUM

Exercise 3 - continued

```
/* Join "customers" table to "orders" table in mycdh
library.
*/
proc SQL;
    CREATE TABLE mycdh.customer_orders
    AS
    SELECT table0.customer_id AS customer_id,
           table0.customer_name AS customer_name,
           table0.customer_state AS customer_state,
           SUM(table1.quantity) AS quantity,
           SUM(table1.total_price) AS net_revenue,
           SUM(table1.total_retail_price) AS gross_revenue,
           COUNT(DISTINCT table1.order_id) AS nbr_orders
    FROM mycdh.customers table0
    INNER JOIN mycdh.orders table1
    ON
       ( table0.customer_id = table1.customer_id )
    GROUP BY table0.customer_id, table0.customer_name,
table0.customer_state
    ORDER BY customer_id
;
quit;
```

1

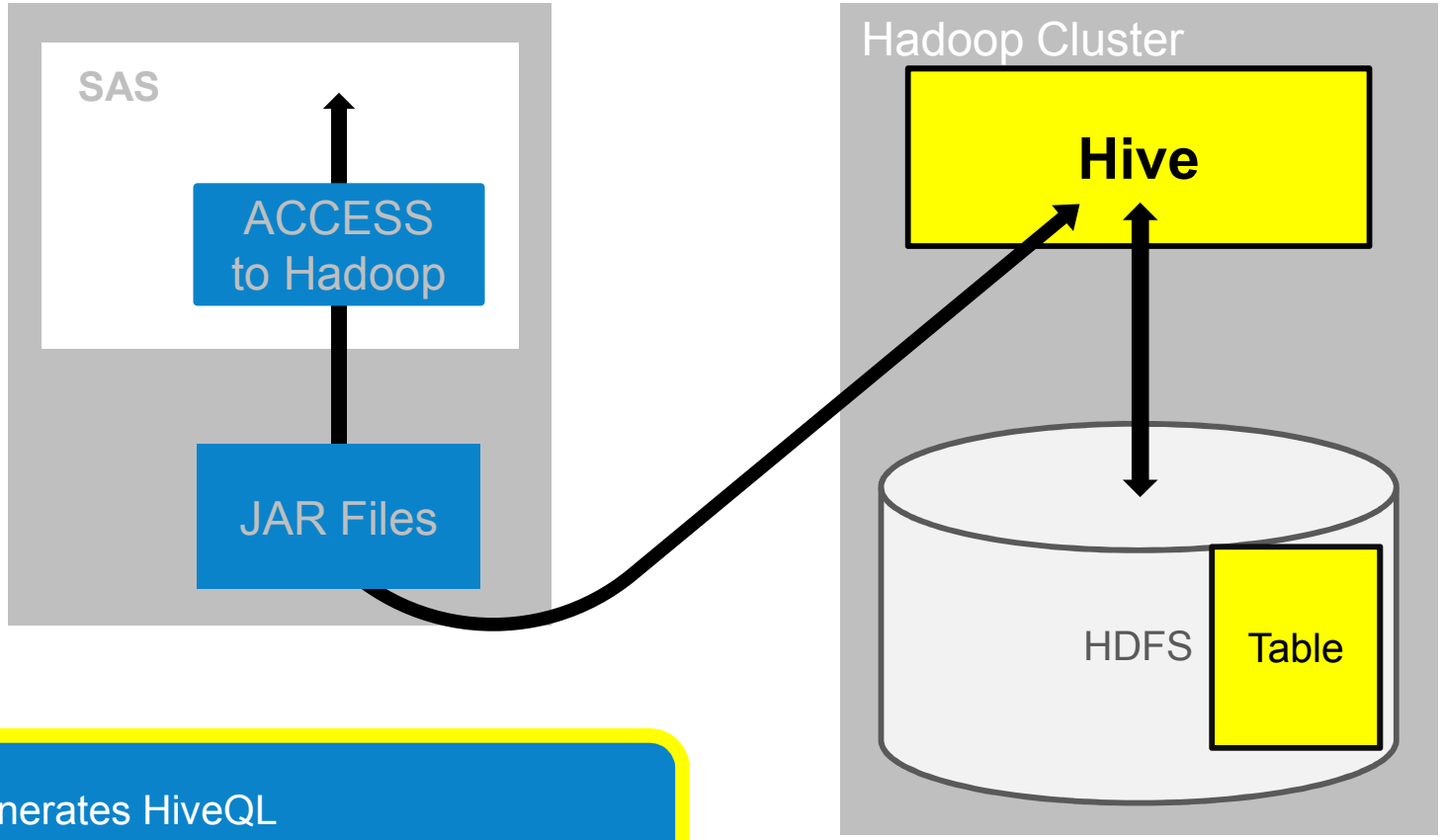
```
/* Join "customers" table from mycdh library to "orders"
table in mycdh2 library.
/* Does it work?
*/
proc SQL;
    CREATE TABLE mycdh.customer_orders2
    AS
    SELECT table0.customer_id AS customer_id,
           table0.customer_name AS customer_name,
           table0.customer_state AS customer_state,
           SUM(table1.quantity) AS quantity,
           SUM(table1.total_price) AS net_revenue,
           SUM(table1.total_retail_price) AS gross_revenue,
           COUNT(DISTINCT table1.order_id) AS nbr_orders
    FROM mycdh.customers table0
    INNER JOIN mycdh2.orders table1
    ON
       ( table0.customer_id = table1.customer_id )
    GROUP BY table0.customer_id, table0.customer_name,
table0.customer_state
    ORDER BY customer_id
;
quit;

/* Clean up after ourselves
*/
proc datasets lib=mycdh; delete customer_orders2; run;
proc datasets lib=mycdh details; run;
```

SAS GLOBAL FORUM

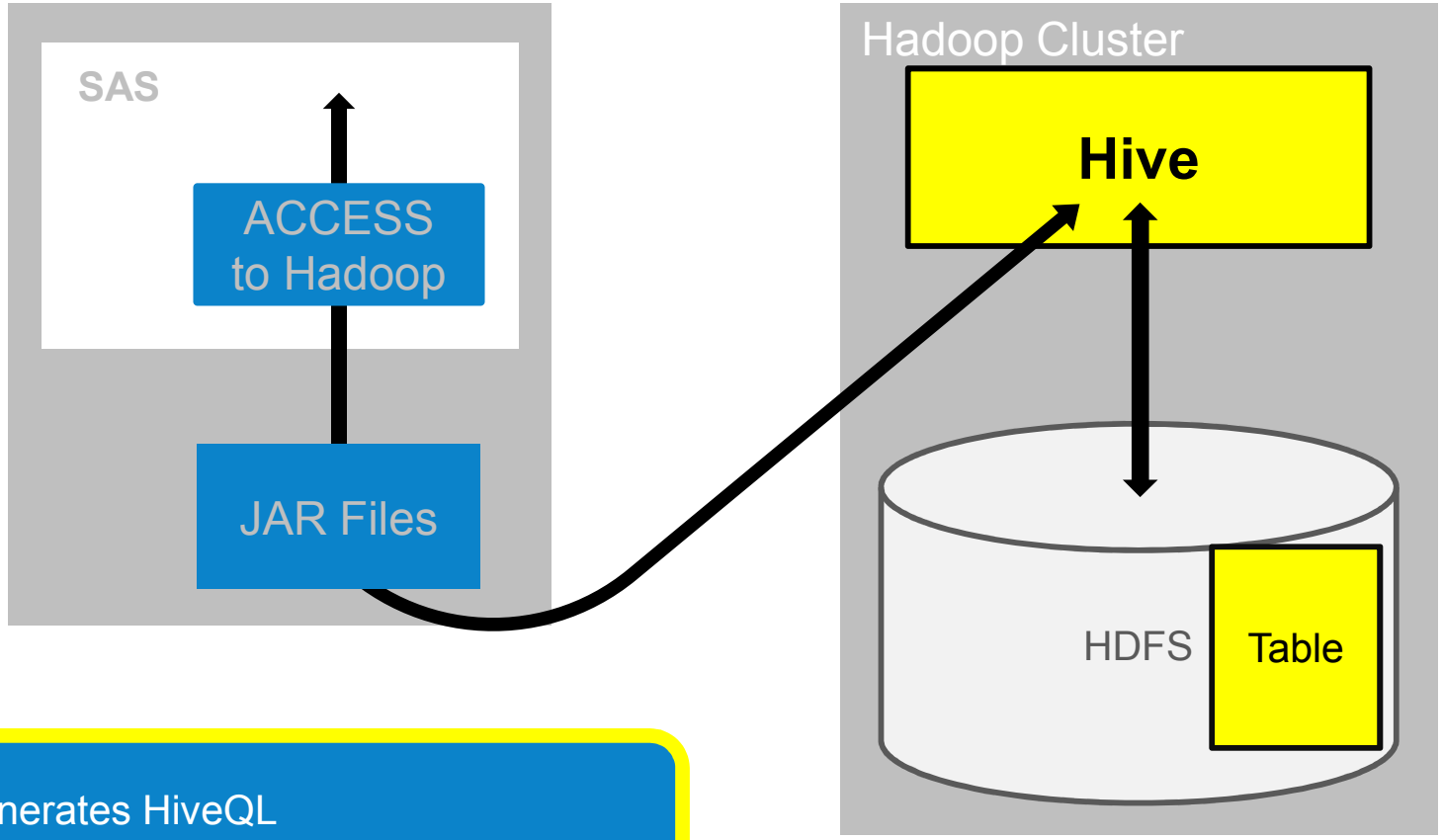
2

SAS Streaming Reads



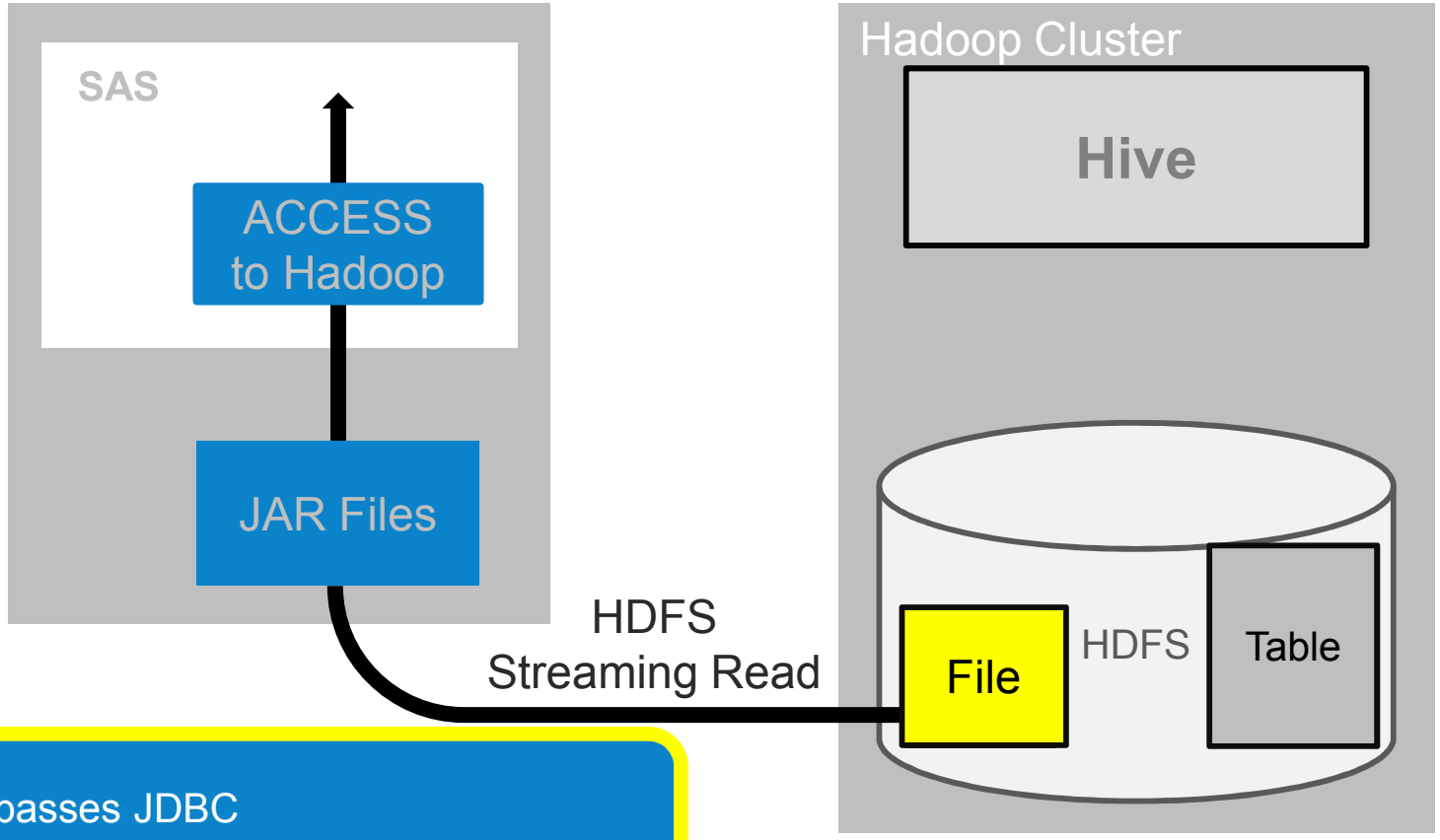
- SAS generates HiveQL
- Subsets the data

SAS Streaming Reads

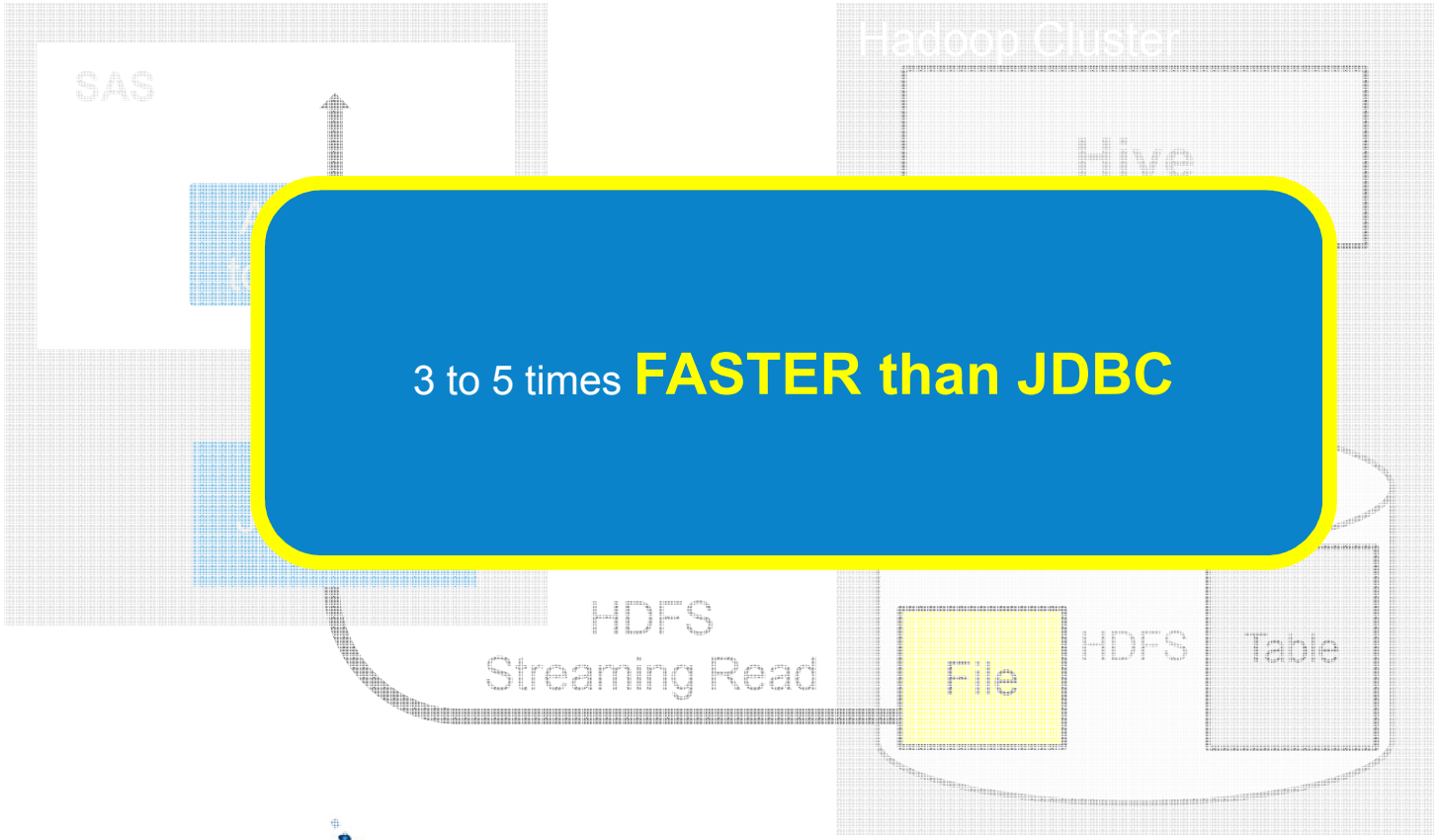


- SAS generates HiveQL
- Subsets the data

SAS Streaming Reads



- SAS bypasses JDBC
- Reads the file directly from HDFS


SAS
Hadoop Cluster
Hive
3 to 5 times FASTER than JDBC
HDFS
Streaming Read
File
HDFS
Table

Explicit Pass-Through

```
PROC SQL;
   CONNECT TO HADOOP(SERVER=cdhsrv
                     PORT=10000);
   EXECUTE(create table testtab
                (col1  string));
   DISCONNECT FROM HADOOP;
QUIT;
```

You write the HiveQL

- CONNECT Statement
- EXECUTE Statement
- DISCONNECT Statement

SAS GLOBAL FORUM

Explicit Pass-Through

```
PROC SQL;
   CONNECT TO HADOOP(SERVER=cdhsrv
                     PORT=10000);
   EXECUTE(create table testtab
               (col1  string));
   DISCONNECT FROM HADOOP;
QUIT;
```

SAS sends this to Hive

You write the HiveQL

SAS GLOBAL FORUM

Explicit Pass-Through

```
proc sql;
   connect to hadoop (server=quickstart
                      user=cloudera);
   execute (create table store_cnt
                 row format delimited
                 fields terminated by '\001'
                 stored as parquet
                as
                   select customer_rk, count(*) as tot
                     from order_fact
                    group by customer_rk) by hadoop;
quit;
```

You write the HiveQL

Exercise 4 – Explicit Passthru

This exercise reinforces the concepts discussed previously.

Exercise 4

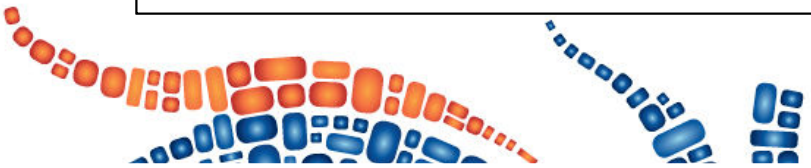
- Open a new program window. Add the following code. Run. Review log.

```
/* CTAS Explicit Pass-thru - all work is handled in Hadoop - execute statement
/* This builds on Exercise 3 - Join
*/
options dbidirectexec;
options nodbidirectexec;

proc sql;
   connect to hadoop (server='quickstart.cloudera' user=cloudera
subprotocol=hive2);

    execute (create table order_cnt
                row format delimited fields terminated by '\001'
                stored as textfile
               as
                  select customer_id, count(*) as total_orders
                    from customer_orders
                   group by customer_id) by hadoop;

   disconnect from hadoop;
quit;
```



Exercise 4 - continued

- Add the following code. Run just this portion (highlight code then run).

```
/* CTAS Implicit Pass-thru */
/* Run it once with NODBIDIRECTEXEC (default) */
/* Run it againg with DBIDIRECTEXEC */

/* Run it with the defaults */
options nodbidirectexec;
proc sql;
   create table mycdh.order_cnt_copy2
       as select *
           from mycdh.customer_orders
          group by customer_id;
quit;

/* optimize it */
options dbidirectexec;
proc sql;
   create table mycdh.order_cnt_copy3
       as select *
           from mycdh.customer_orders
          group by customer_id;
quit;
```

Exercise 4 - continued

- Add the code on the left. Run just this portion (highlight code then run).
- Run the code on the right.

```
/* Create a SAS data set from Hadoop data */
/* Does the order of the join tables matter? */
proc sql;
   create table work.join_test as (
      select c.customer_id, o.product_id
        from mycdh.customers c
           , mycdh.orders o
       where c.customer_id = o.customer_id);
quit;

/* PROC FREQ example */
data mycdh.cloudera_class;
   set sashelp.class;
run;

proc freq data=mycdh.cloudera_class;
   tables sex * age;
   where age > 9;
   title 'Catchy Title Goes Here';
run;
```

```
/* Clean up */
proc sql;
   connect to hadoop (server='quickstart.cloudera'
   user=cloudera subprotocol=hive2);

   execute (drop table order_cnt) by hadoop;
   execute (drop table order_cnt_copy2) by hadoop;
   execute (drop table order_cnt_copy3) by hadoop;
   execute (drop table cloudera_class) by hadoop;

   drop table work.join_test;

   disconnect from hadoop;

quit;
```

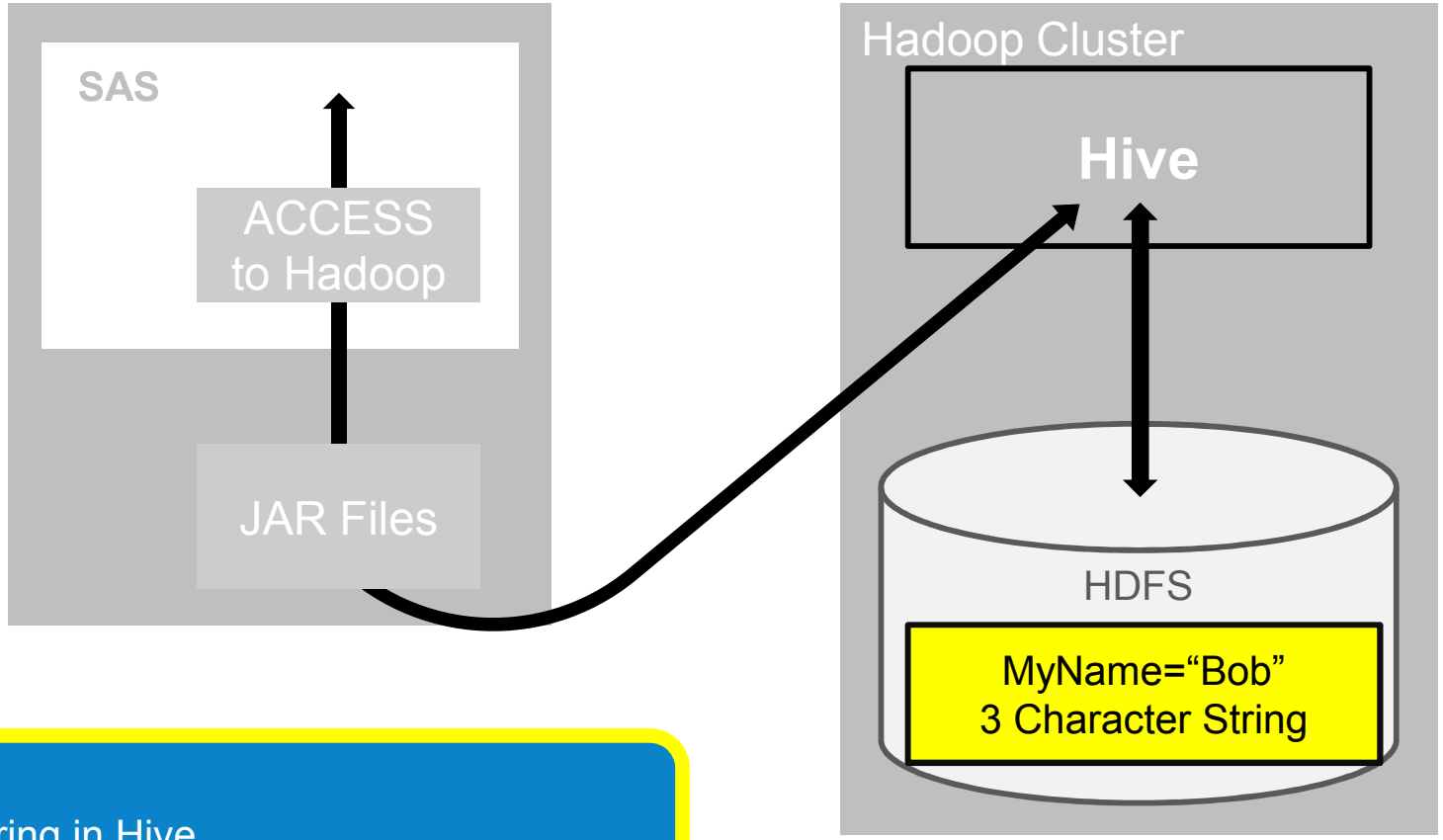
SAS GLOBAL FORUM

The 32K String Thing

cloudera

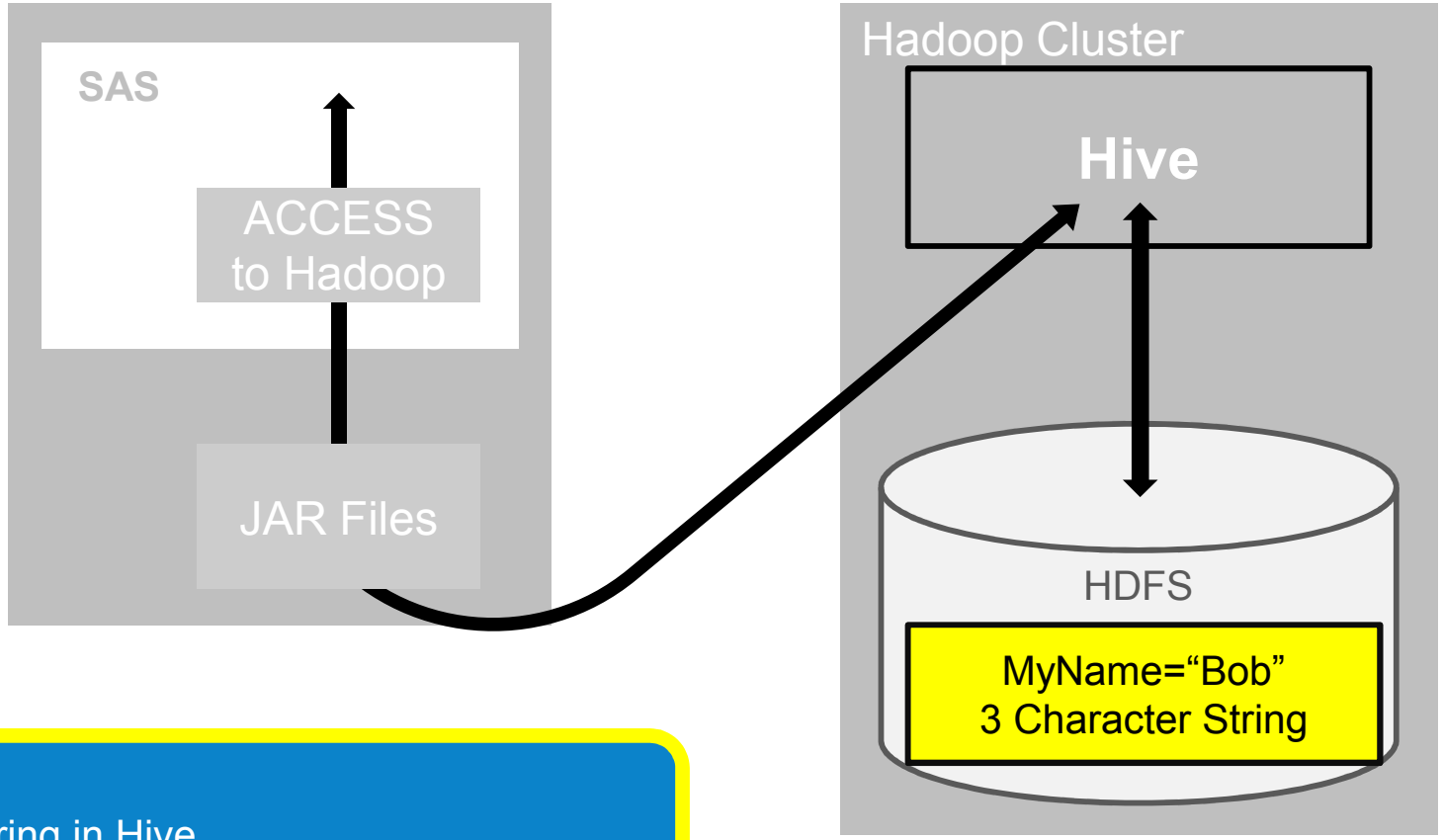
sas

Reading Java Strings from Hive



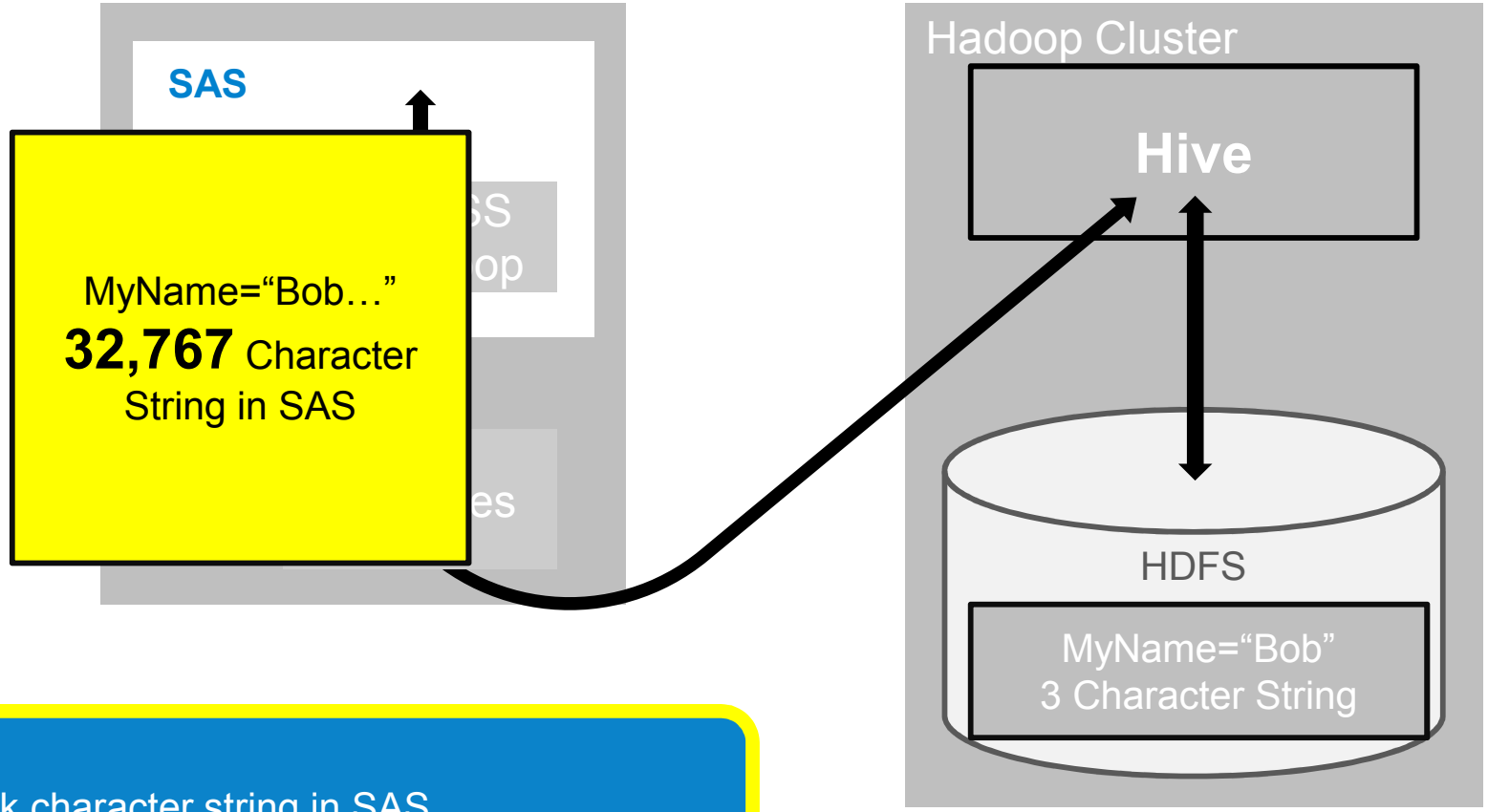
- Java String in Hive

Reading Java Strings from Hive



- Java String in Hive

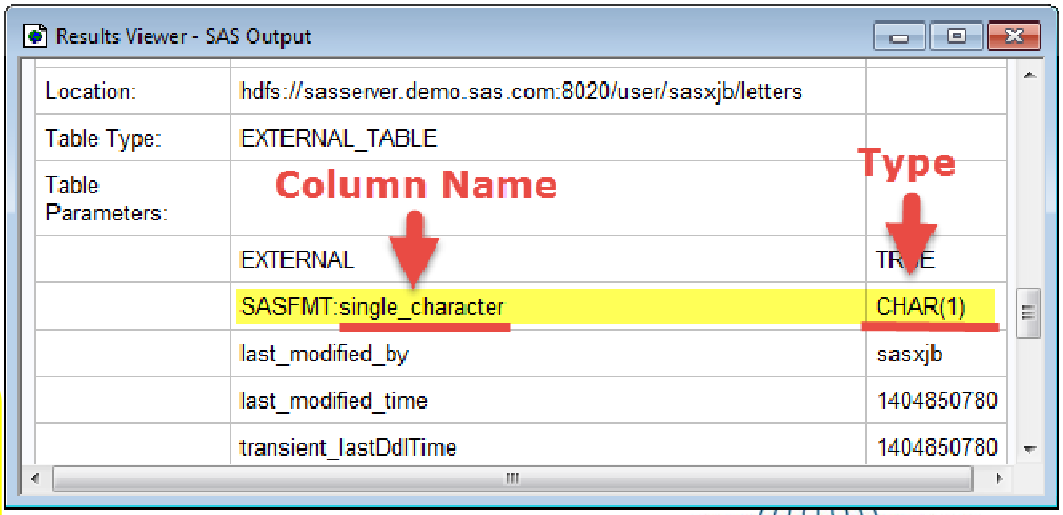
Reading Java Strings from Hive



- 32k character string in SAS

What is a Table Attribute?

```
proc sql;
   connect to hadoop (server=cloudera44
                      subprotocol=hive2);
   select * from connection to hadoop
      (DESCRIBE FORMATTED letters);
quit;
```



The attribute can “fix” the 32K problem

How Can You Set a Table Attribute?

CREATE TABLE Using SAS Implicit Pass-Thru

```
data mycdh.letters;
   set work.letters; /* one column; one character */
quit;
```

ALTER TABLE

```
/* Assume table created outside of SAS */
proc sql;
   connect to hadoop (…connect info here…);
   execute (alter table letters set tblproperties
         ('SASFMT:single_character'='CHAR(1)')) by hadoop;
quit;
```

SAS GLOBAL FORUM

When Will SAS Use a Table Attribute?

When IMPLICIT PASS-THRU is used

```
data work.letters;
   set mycdh.letters;
quit;
```

And table attributes are defined on the table

SAS GLOBAL FORUM

Exercise 5 – Strings

This exercise reinforces the concepts discussed previously.

Exercise 5

- Copy file “single_string_column.txt” to your desktop.
- Open a new program window and add the following code. Run.

Change this to your local userid

```
/* setup the environment  */
filename cfg 'C:\Program Files\hadoop\conf\combined-site.xml';

/* Copy single_string_column.txt to HDFS. */
proc hadoop options=cfg username="cloudera" verbose;
   HDFS COPYFROMLOCAL="C:\Users\<userid>\Desktop\single_string_column.txt"
                      OUT='/user/cloudera/letters/single_string_column.txt';
run;

proc sql;
   connect to hadoop (server='quickstart.cloudera' user=cloudera subprotocol=hive2
   schema='default');

    execute (create external table letters (single_character string)
             row format delimited fields terminated by ','
             location '/user/cloudera/letters';) by hadoop;
   disconnect from hadoop;
quit;
```

SAS GLOBAL FORUM

Exercise 5 - continued

- Add the following code. Run.

```
/* Create a SAS data set from letters */
data work.letters;
   set mycdh.letters;
run;
```

- Check the table properties. What is the size of the column?

Exercise 5 - continued

- Add the following code and run just this portion.

```
/* Let's fix it */
proc sql;
   connect to hadoop (server='quickstart.cloudera'
   user=cloudera subprotocol=hive2 schema='default');
   execute (alter table letters set tblproperties

   ('SASFMT:single_character'='CHAR(1)')) by hadoop;
   disconnect from hadoop;
quit;

/* Did the ALTER TABLE fix it? */
data work.letters_fixed;
   set mycdh.letters;
run;z
```

1

```
/* What about explicit pass-thru? */
proc sql;
   connect to hadoop (server='quickstart.cloudera'
   user=cloudera subprotocol=hive2 schema='default');
   create table work.letters_ep as
      select * from connection to hadoop
         (select single_character from letters);
   disconnect from hadoop;
quit;

/* How can you deal with explicit pass-thru */
proc sql;
   connect to hadoop (server='quickstart.cloudera'
   user=cloudera subprotocol=hive2 schema='default');
   create table work.letters_ep_fixed as
      select single_character format $1.
        from connection to hadoop
           (select single_character from letters);
   disconnect from hadoop;
quit;
```

2

Exercise 5 - continued

- Add the following code and run just this portion.

```
/* How can you see the table properties defined on a
table? */
proc sql;
   connect to hadoop (server='quickstart.cloudera'
   user=cloudera subprotocol=hive2 schema='default');
   select * from connection to hadoop
      (describe extended letters);
   disconnect from hadoop;
quit;
```

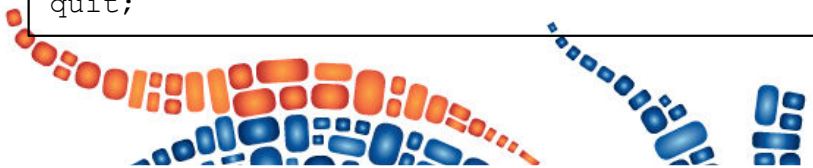
1

```
/* Is there a better way? */
proc sql;
   connect to hadoop (server='quickstart.cloudera'
   user=cloudera subprotocol=hive2 schema='default');
   select * from connection to hadoop
      (describe formatted letters);
   disconnect from hadoop;
quit;
```

2

- Add the following code to clean up. Run just this portion.

```
/* This will drop the outBook directory and all the contents */
/* Useful for cleaning up after running the MapReduce */
proc hadoop options=cfg username="cloudera" verbose;
   HDFS delete='/user/cloudera/letters/single_string_column.txt';
run;
proc sql;
   connect to hadoop (server='quickstart.cloudera' user=cloudera
subprotocol=hive2 schema='default');
   execute (drop table letters;) by hadoop;
   disconnect from hadoop;
quit;
```

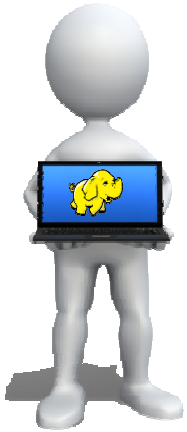


Solution Overview

SAS® Data Loader for Hadoop

SAS GLOBAL FORUM

SAS Data Loader 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 Hadoop
Query a table, or join data from multiple tables

Sort and De-Duplicate Data i...
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 Hadoop
Transform data from a Hadoop table

Transpose Data in Hadoop
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 replaced

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

Copyright © 2015, SAS Institute Inc. All rights reserved.

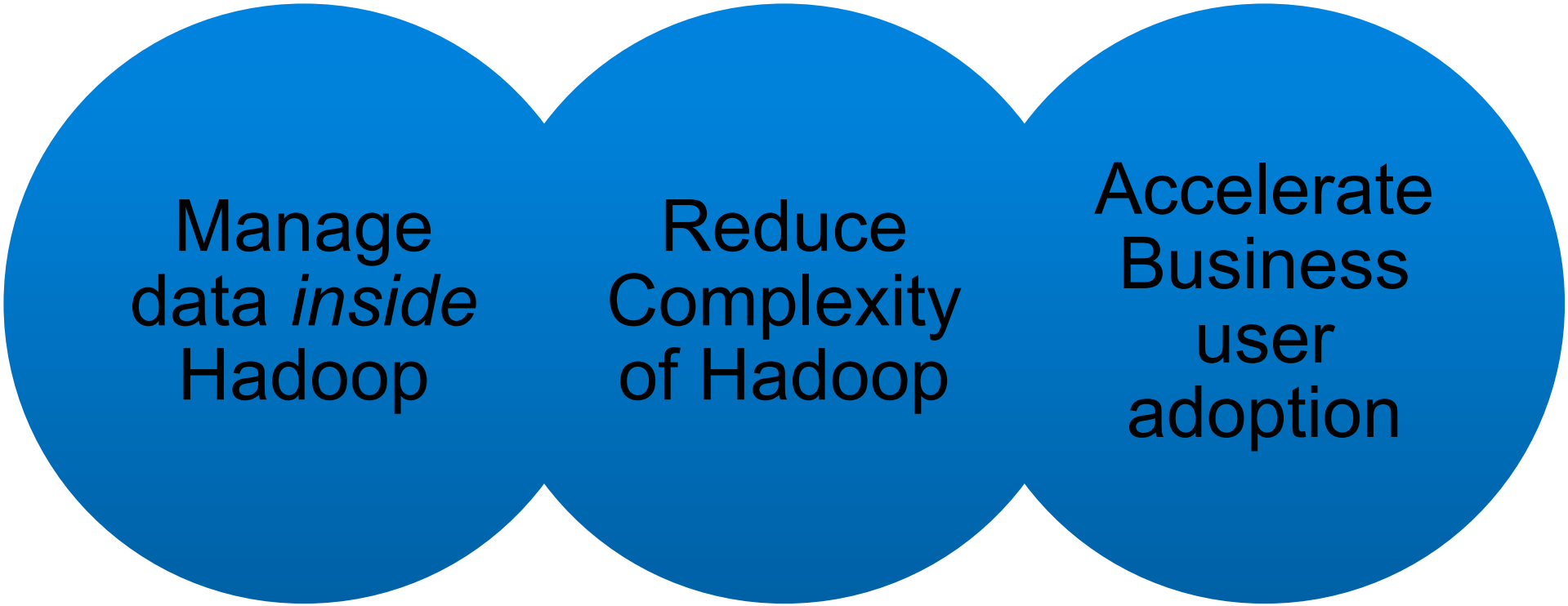
SAS Data Loader for Hadoop…

"Purpose-built" easy to use data management solution to specifically address: acquiring, structuring, cleaning and transforming data inside Hadoop

SAS Data Loader for Hadoop is a smart approach, turning the Hadoop environment into a productive environment; where barriers are removed, and data is accessible and usable

Copyright © 2015, SAS Institute Inc. All rights reserved.

SAS Data Loader enables organizations to…



Copyright © 2015, SAS Institute Inc. All rights reserved.

SAS Data Loader for Hadoop

Use Cases

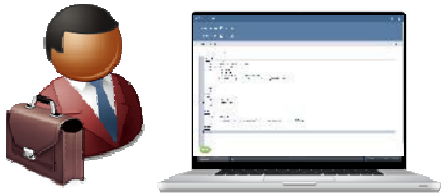
SAS® Data Loader for Hadoop

I need my Customer data in Hadoop

Copy Data to Hadoop
Copy data from a database into Hadoop

I need to subset and summarize the data

Transform Data in Hadoop
Transform data from a Hadoop table



I can see, but I can also fix, the data quality issue

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

I need to Load the data to LASR for visualization

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

SAS® Data Loader - Trial

Hands-On Exercises

Copy Data to Hadoop

1. First, we must import the source datasets that were downloaded, and placed in the virtual machine shared directory. To do this, use your web browser to **open SAS Data Loader** as described in the SAS Data Loader for Hadoop installation and configuration instructions.
2. Select, the **Copy Data to Hadoop** icon from the main menu.
3. For the Source Table, select **SAS Server**, then select **Sample Data**. This will use the local SAS session on the VM to locate the SAS datasets that you placed in your shared virtual machine directory.
4. There you will see the several datasets that were copied to your shared virtual machine directory' Select the first '**CUSTOMERS**' dataset, and select **Next**.
5. We will be pulling the entire table in, so you may click **Next** on the default selections on the next two steps in order to include **All Rows** and **All Columns**.
6. For the Target Table, select the **default schema**, and select the **New Table…** option. Name your new table, "**customers**," and select **Next**.
7. Review the code section provided, then select **Next**.
8. Select, **Start copying data**. Once the copy is finished as indicated by the message, "**Successfully copied data**," and you may **View the Results**.
9. Repeat steps 2-8 for the '**ORDERS'** table by selecting the Back to Directives option.

Profile Data In Hadoop

1. Now we must profile the data to determine the quality. To do this, go to the main page by selecting "**Back to Directives**" and select **Profile Data**.
2. Select a base table by navigating to **default** schema, and select **customers**.
3. Here you can select all or a subset of columns. We'll run the profile on all columns by selecting **Next**.
4. Give a Report Name by typing in **Customer Profile Report PreCleanse** and select **Next**.
5. Run the report by selecting **Create Profile Report**.
6. Once the report completes, select **View Profile**. Note on the report that there are no nulls or blank values. This is important when joining additional tables as null values on a join column could cause a Cartesian product.
7. At the top of the page select **Show Outline** and click on the column for **customer_address**. Note that the address data under **Frequency Distribution** has differing case which calls for the need to standardize the field values.
8. Next, click on **customer_city** from the list of columns. Note that the address data under **Frequency Distribution** has differing case which calls for the need to standardize the field values.
    a. Do we have multiple occurrences of the same city?
9. Finally, click on **customer_state** from the list of columns.
    a. How many states are there?
    b. Do states appear in both full name and abbreviation?
10. **Frequency Distribution** has differing case which calls for the need to standardize the field values.

Cleanse Data In Hadoop

1. Now that we understand which values in our customers table require cleansing, let's initiate a transformation. To do this, go to the main page by selecting "**Back to Directives**" and select **Cleanse Data in Hadoop**.
2. Select a base table by navigating to the **default** schema and select the **customers** table. Once the table is selected click **Next**.
3. On the Cleanse Data in Hadoop, select the **Standardize Data** directive.
4. Under C**olumn**, select the **customer_address** column.
5. Under **Definition**, select the **Address** definition. Note that Data Loader provides a default New Column Name and Character limit which the user can edit if required.
6. Select **Add Column**.
7. Under **Column**, select the **customer_city** column.
8. Under **Definition**, select the **City** definition.
9. Under C**olumn**, select the **customer_state** column.
10. Under **Definition**, note that Data Loader provides a both State/Province (Abbreviation) and (Full Name). Select the **State/Province (Full Name)** definition. .

Cleanse Data In Hadoop - continued

11. Select **Next**.
12. Choose the **default** schema and select **New Table**. Provide the table name **customers_standardize** and select **Next**.
13. Finally, select **Start transforming** data once the process complete select **View the Results**. Here we can view the differences between the columns by selecting **customer_address, customer_address_standardized, customer_city, customer_city_standardized, customer_state and customer_state_standardized.**

Join Data From Multiple Tables

1. In this section we will join the tables we just imported together using a common column name. This is done by selecting the **Query or Join Data in Hadoop** icon.
2. Select a base table by navigating to the **default** schema, and select **orders** table. Once loaded, select **Add Join**.
3. On the join selection under **Choose a table**, locate the **customers_standardize** table, and make sure the two tables are using the **customer_id** column on which to join.
4. Select **Next**.
5. We create an aggregation on this table by selecting the **product_id** under **Group rows by**.
6. Next, select **Add column** and choose the **customer_state_standardized**.
7. Under Summarize columns choose **total_retail_price** and then select **Sum** under **Aggregation**. Accept the default name under **New column name**. Select **Next**.
8. Select **Next** to accept **All rows**.
9. Select **Next** on the **Columns specified in "Summarize Rows"**
10. Select **Next** on the sort step to accept the default.

Join Data From Multiple Tables - continued

11. For the Target Table, select the **default schema**, and select the **New Table…** option. Name your new table, "**orders_joined**," and select **Next**.
12. Review the code section provided, then select **Next**.
13. Select, **Start joining data**. Once the join is finished as indicated by the message, "**Successfully copied data**," and you may **View the Results**.

SAS GLOBAL FORUM

Transpose Data in Hadoop

1. Transposing data allows us to pivot rows to columns or columns to rows for further analytical processing. Under the main page, select **Transpose Data in Hadoop**.
2. On the right side of the screen are descriptions and visuals of the various roles under the transpose function.
3. Select the **total_retail_price_sum** column from **Columns**, highlight **Transpose columns** by selecting it and finally click ➪ to add the column value under roles.
4. Select the **customer_state_standardized** column from **Columns**, highlight **Columns to group by** by selecting it and finally click ➪ to add the column value under roles.
5. Select the **product_id** column from **Columns**, highlight **ID columns** by selecting it and finally click ➪ to add the column value under roles.
6. Choose **Next**.
7. Select **default** schema and choose **New Table**. Provide the name **orders_analytic_base_table** and select **OK**.
8. Choose **Next**.
9. Select **Start transposing data**. When the process is complete you may **View the Results**.

SAS GLOBAL FORUM

SAS® GLOBAL FORUM

April 26-29
Dallas, TX