



SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

An Insider's Guide to SAS/ACCESS to Hadoop

Jeff Bailey, SAS

#SASGF



About the presenter


Jeff Bailey is a Principal Product Manager in the Data Management group in SAS Product Management.

Jeff has been with SAS since 1992 and has spent most of his time helping customers use SAS with database management systems.

He has held positions in SAS R&D, Consulting, Education and Product Management.

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?
- 

The Workshop Environment

- SAS 9.4M3
- SAS/ACCESS Interface to Hadoop
- Cloudera Quickstart VM (CDH 5.5)

```
C:\HOW\bailey  
-- \code  
-- \data  
-- \exercises  
-- \insights  
-- \saslogs
```

This Workshop Will Be Different...

Exercises



Discussions

~~LECTURE~~

If You Don't Feel Like Running the Examples...

C:\HOW\bailey\saslogs



EXERCISE 0

Test the Environment

Start the CDH 5.5 Environment

- Boot the virtual machine
- Start SAS
- Issues a LIBNAME Statement
- Hope for the best!

Submit this LIBNAME Statement

```
LIBNAME mycdh HADOOP SERVER='quickstart.cloudera'  
          USER='cloudera'  
          PASSWORD='cloudera' ;
```

Understand: The SAS and CDH environments have been configured to work. This is not a trick.

The workshop environment is designed for functionality not performance.

The VM is fraught with peril... Sometimes it doesn't work well... If this happens we will have to improvise...

We Have a Plan B, and C, ...

C:\HOW\bailey\saslogs

Environment Details

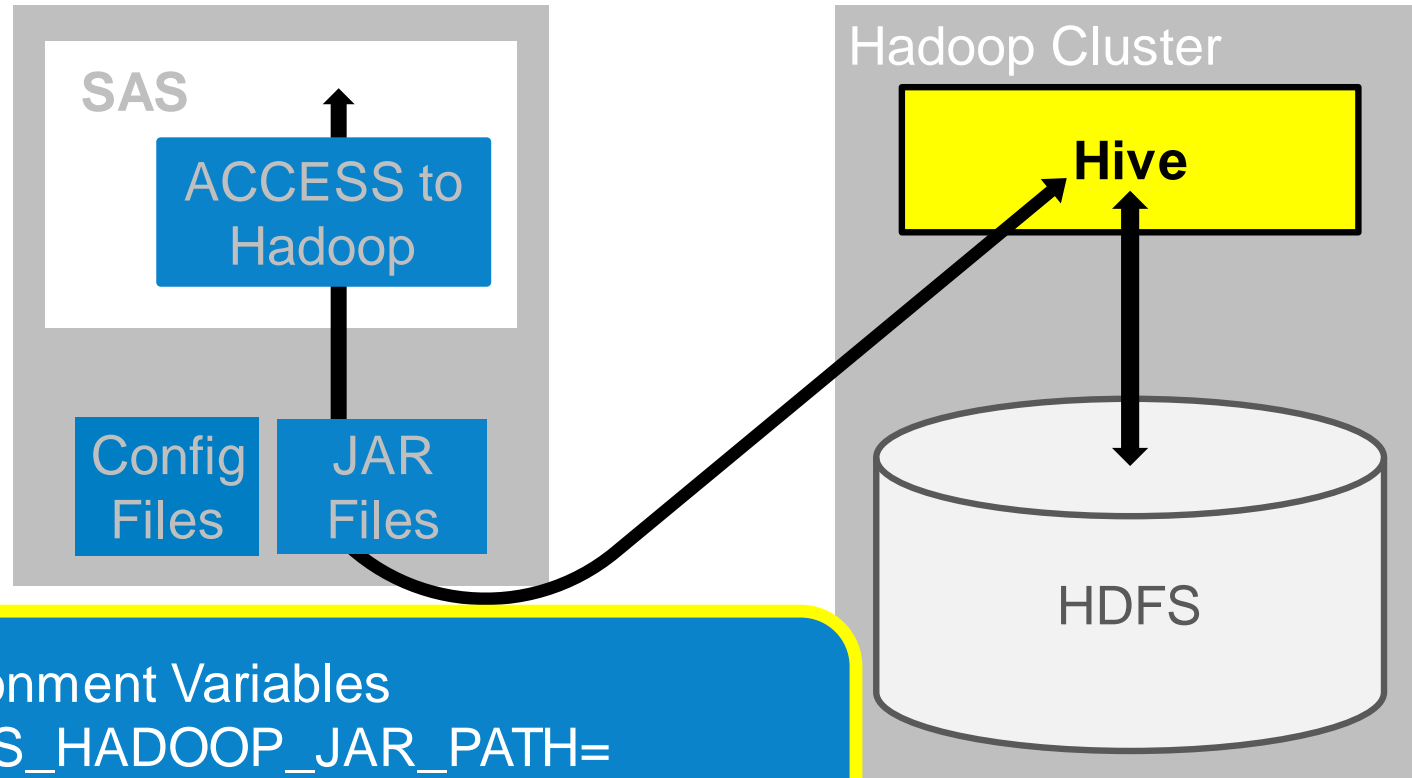
- Libname mycdh Hadoop server='quickstart.cloudera user=cloudera password=cloudera;
- Ping quickstart.cloudera
- Open browser on the desktop:
<http://quickstart.cloudera:7180>
 - Logon cloudera/cloudera
- Open HUE <http://quickstart.cloudera:8888>
 - Logon cloudera/cloudera



EXERCISE 1

ARCHITECTURE

HOW DOES SAS CONNECT TO HADOOP?



Environment Variables

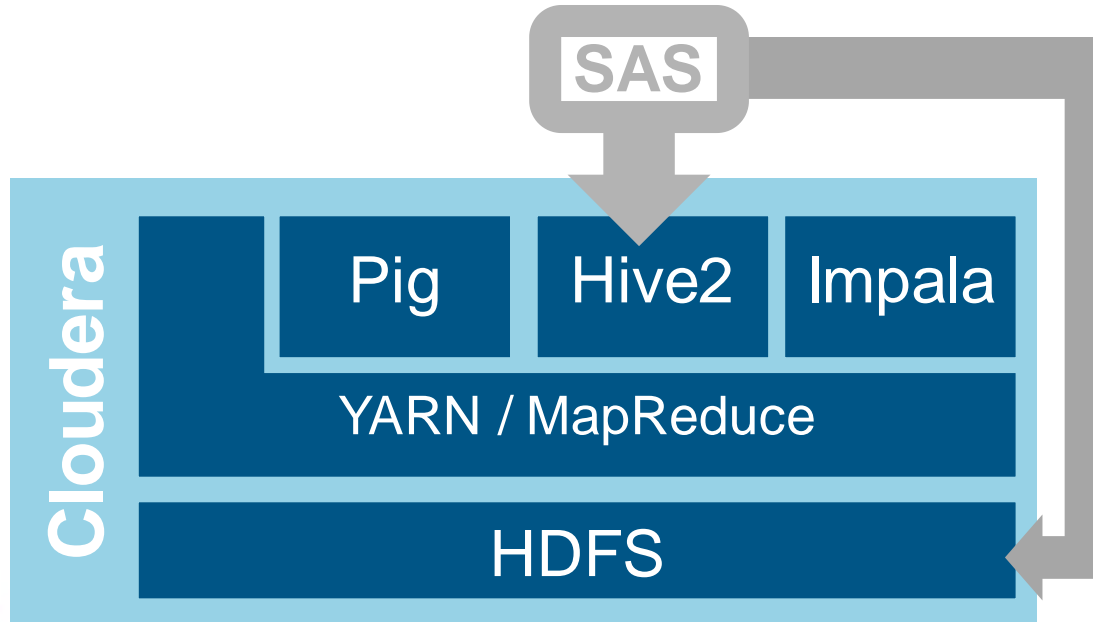
- SAS_HADOOP_JAR_PATH=
- SAS_HADOOP_CONFIG_PATH=
- SAS_HADOOP_RESTFUL=

You Can Set Env Variables from SAS

- If env variables are set at the OS run the OPTIONS statements prior to running SAS code
- If your env changes don't take effect, restart SAS

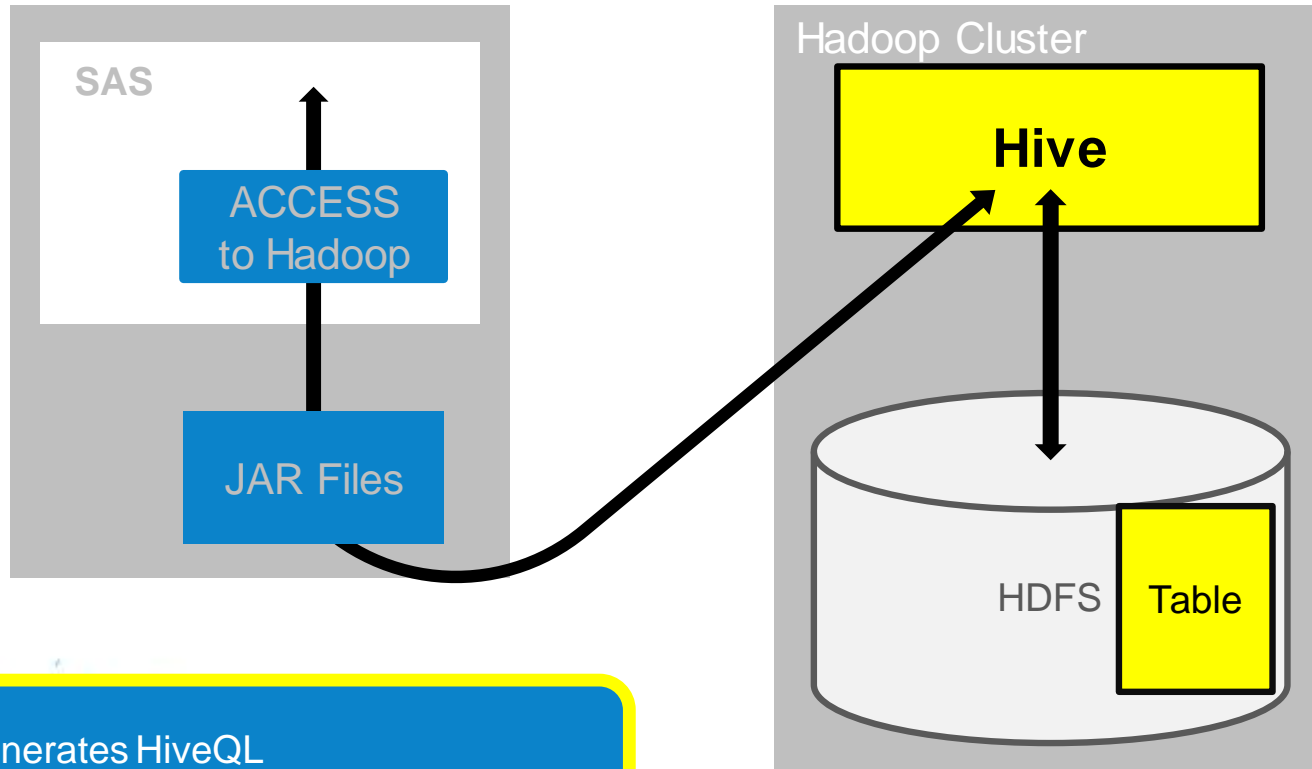
```
OPTIONS SET=SAS_HADOOP_JAR_PATH="C:\CDH_JARS";  
OPTIONS SET=SAS_HADOOP_CONFIG_PATH="C:\CDH_CONFIG";  
OPTIONS SET=SAS_HADOOP_RESTFUL=1;
```

SAS/ACCESS TO HADOOP



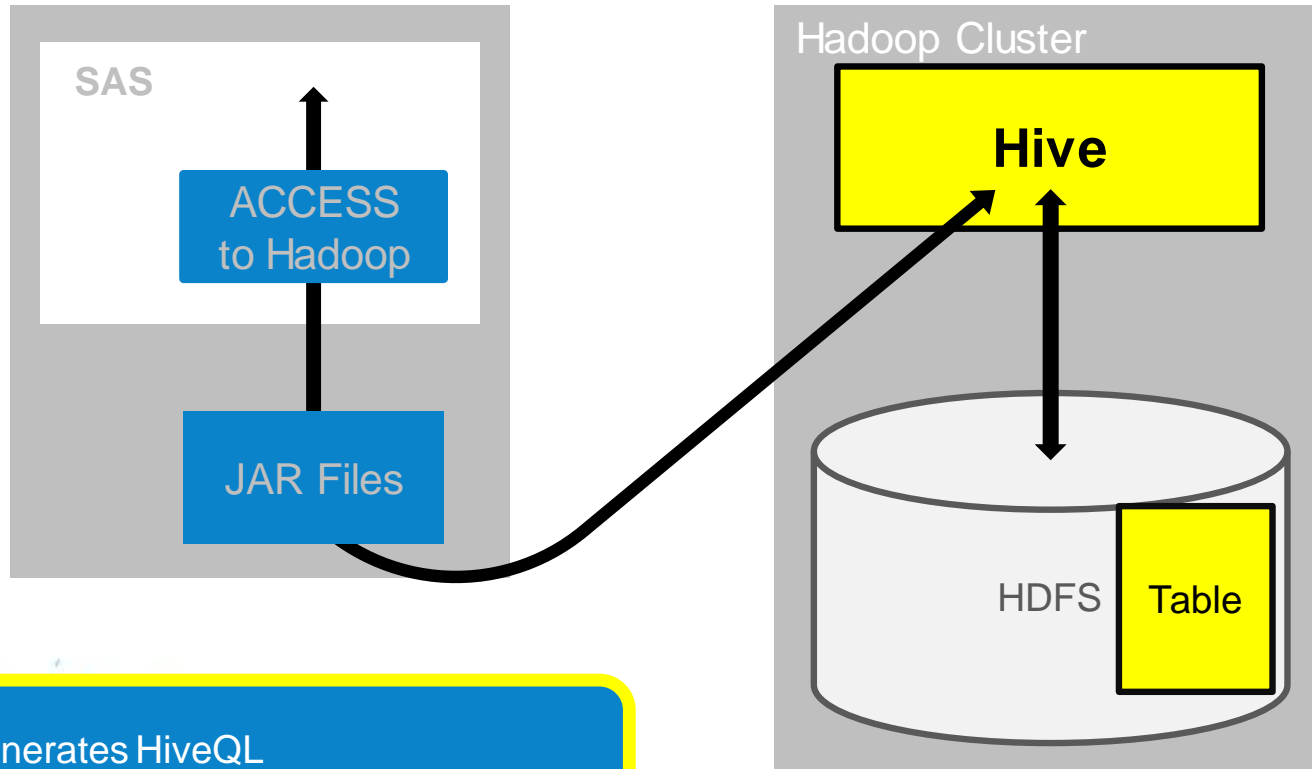
- Connects via JDBC
- Makes Hive tables look like SAS data sets
- Converts SAS code to HiveQL
- Can use webHDFS
- Bulk loads directly to HDFS

SAS Streaming Reads



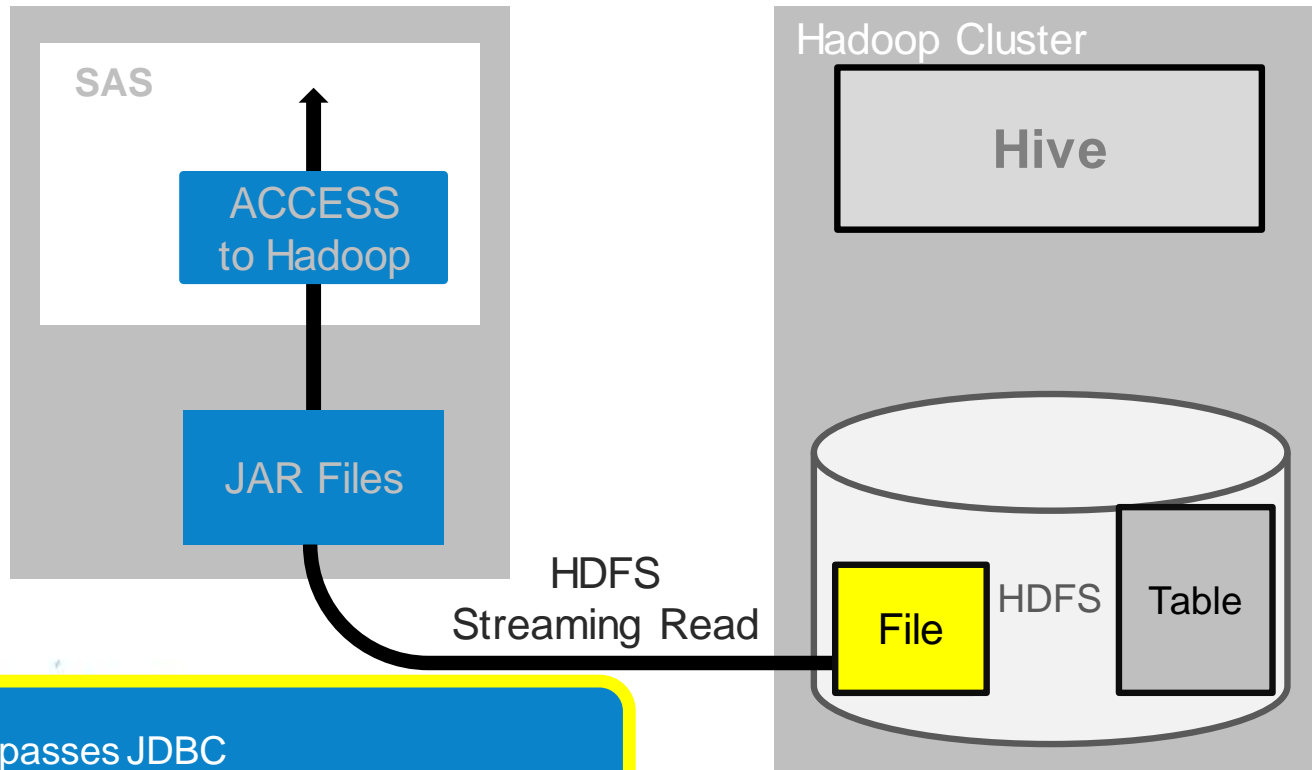
- SAS generates HiveQL
- Subsets the data

SAS Streaming Reads

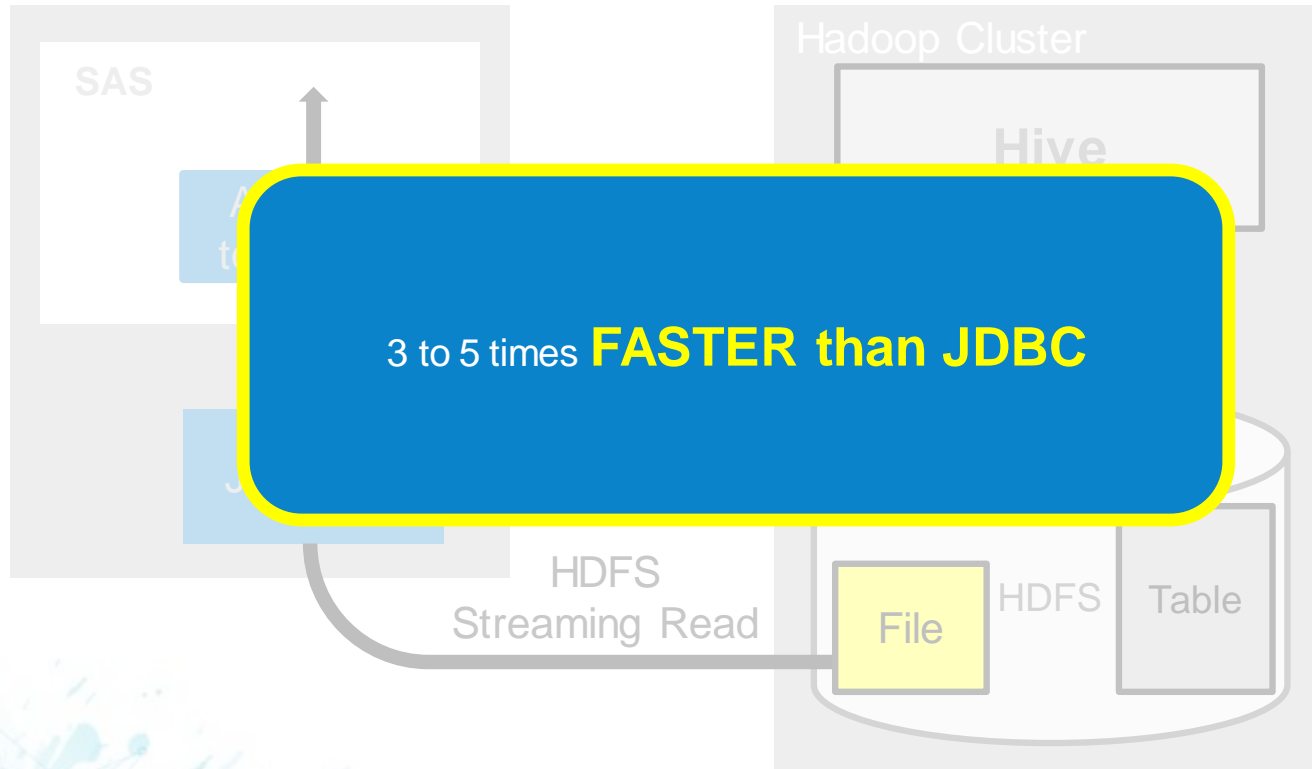


- SAS generates HiveQL
- Subsets the data

SAS Streaming Reads



- SAS bypasses JDBC
- Reads the file directly from HDFS



EX 01 – Architecture Insights

- JDBC based
- SAS/ACCESS Interface to Hadoop requires “some” configuration.
- Must set environment variables
 - SAS_HADOOP_CONFIG_PATH=
 - SAS_HADOOP_JAR_PATH=
- SAS_HADOOP_RESTFUL=1 eases the configuration burden
- SAS Deployment Manager eases the configuration burden
 - Requires Oozie



EXERCISE 2

Tables and Files

Passing Hadoop Configuration Parameters

```
LIBNAME mycdh HADOOP SERVER=cdhserver
```

```
  PROPERTIES="mapreduce.map.memory.mb=2048";
```

How Does SAS/ACCESS Talk to Hadoop?

```
proc sql;  
    select count(*) from mycdh.customer_dim  
        where loyalty_program='Chocolate Club';  
run;
```

?

SAS/ACCESS Uses SQL to Talk to Hive2

```
proc sql;  
    select count(*) from mycdh.customer_dim  
        where loyalty_program='Chocolate Club';  
run;
```

```
select COUNT(*) from `CUSTOMER_DIM` TXT_1  
WHERE TXT_1.`loyalty_program` = 'Chocolate Club'
```

SAS Generated This SQL

Use the SAS OPTIONS Statement to See SQL

```
proc sql;  
    select count(*) from mycdh.customer_dim  
        where loyalty_program='Chocolate Club';  
run;
```

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

```
select COUNT(*) from `CUSTOMER_DIM` TXT_1  
WHERE TXT_1.`loyalty_program` = 'Chocolate Club'
```

SAS Generated This SQL

Understanding SASTRACE= Output

HADOOP_53: Prepared: on connection 2
SHOW TABLES 'CUSTOMER_DIM'

Does the table exist?

HADOOP_54: Prepared: on connection 2
DESCRIBE FORMATTED CUSTOMER_DIM

Get the extended table attributes

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

Get the column information

HADOOP_56: Executed: on connection 3
USE `default`

HADOOP_57: Prepared: on connection 3
select COUNT(*) from `CUSTOMER_DIM` TXT_1
where TXT_1.`loyalty_program` = 'Chocolate Club
Member'

Execute the SQL code

HADOOP_58: Executed: on connection 3
select COUNT(*) from `CUSTOMER_DIM` TXT_1 where
TXT_1.`loyalty_program` = 'Chocolate Club Member'

Does nothing. Listed for
consistency with other engines

Explicit Pass-Through

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

You write the HiveQL

- CONNECT Statement
- EXECUTE Statement
- DISCONNECT Statement

Explicit Pass-Through

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

SAS sends this to Hive

You write the HiveQL

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

EX 02 – Tables and Files Insights

- MapReduce should be avoided if at all possible.
- You can delete a table and leave the file - External Tables.
- You can delete a file and leave the table - this helps with ETL and other things. It makes the environment very flexible.
- It is possible to create a table which does not match the underlying data. This may result in errors or unexpected results. The data is validated against its definition when it is read - SCHEMA-ON-READ.
- Data base management systems use SCHEMA-ON-WRITE. This means that data that does not match the defined data type cannot be written to the database - SCHEMA-ON-WRITE.

EX 02 – Tables and Files Insights (con't)

- PROC HADOOP - is a great housekeeper. You can use it to clean-up after jobs. It is also very useful for setting up HDFS directories, etc.
- HUE (Hadoop Users Experience) is a great way to learn what SAS is doing. It is also a great tool for doing real work - like finding files and working on queries.



EXERCISE 3

JOINS and CTAS

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?

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



Add CTAS slides here...

EX 03 – JOIN and CTAS Insights

- It is easy to code examples to test whether X processing happens inside the database.
- If you use any SAS/ACCESS product this is one of the most important things that you can know. You should memorize this OPTIONS statement:
- `options sastrace=',,,d' sastraceloc=saslog nostsuffix;`
- The most interesting thing here is that using two different hostnames, which point to the same server, will cause a JOIN to be post processed in SAS. I guess this shouldn't surprise me, but it does.

EX 03 – JOIN and CTAS Insights

- Similarly, if you include USER= and/or PASSWORD= on only one LIBNAME statement the JOIN will not be passed to Hive.
- Cross schema JOINS are not allowed here. In other ACCESS products they are allowed. Know your SAS/ACCESS product.
- CREATE TABLE AS is very expensive if it is not passed to Hive. You do this you should ensure that Hive handles the processing. Look up DBIDIRECTEXEC and make sure you understand it.
- Note the use of EXPLICIT PASS-THRU to create/drop Hive schemas. It is a great example of the types of things you should use it for.



EXERCISE 4

BULK LOAD

EX 04 – Bulk Loading Insights...

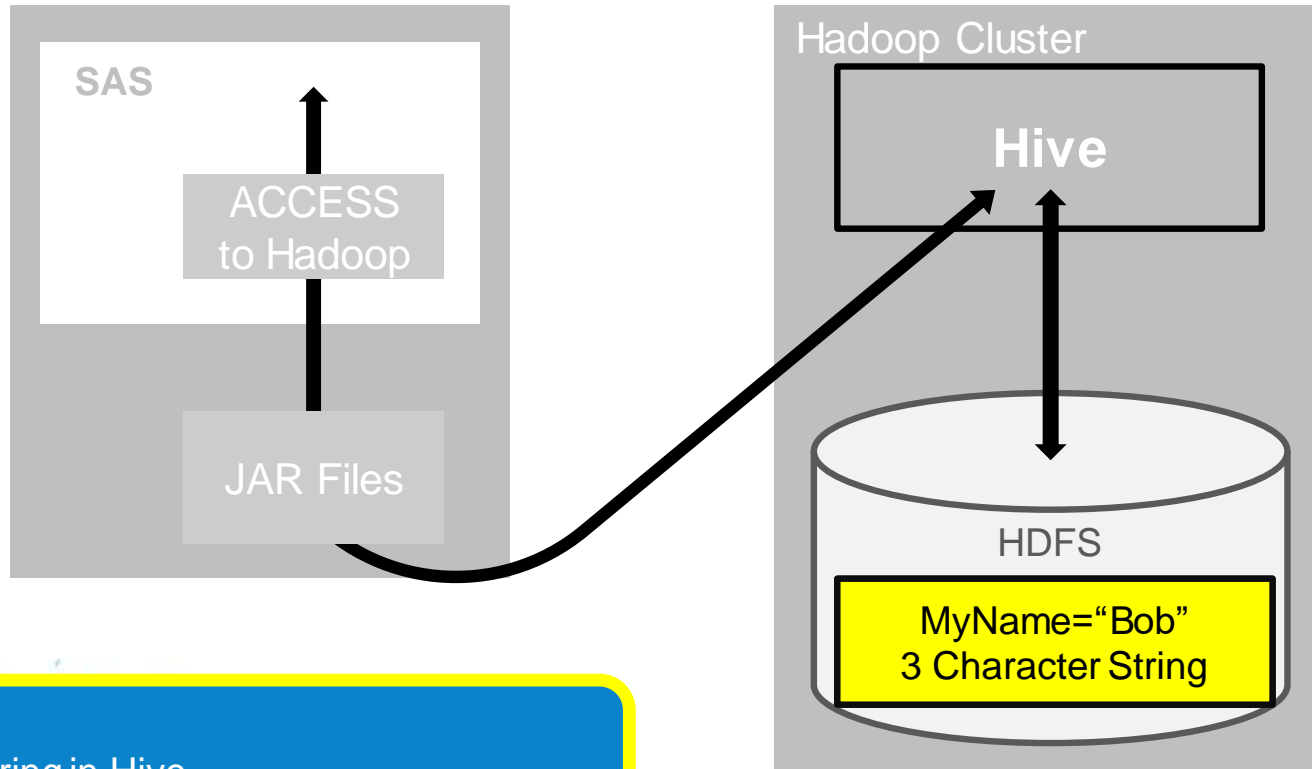
- Anytime you write data to Hive using SAS/ACCESS to Hadoop you are using bulk load. This means that it doesn't matter if you set BULKLOAD=. For example, setting BULKLOAD=NO means that you are actually bulk loading.
- This requires configuration:
 - SAS_HADOOP_CONFIG_PATH=
 - SAS_HADOOP_JAR_PATH=
 - SAS_HADOOP_RESTFUL=1 - Highly recommended: eases the JAR configuration.
- SAS/ACCESS to Hadoop actually bypasses JDBC and reads/writes directly to HDFS. This greatly increases performance.
- You can short circuit the bypassing JDBC for reading by setting READ_METHOD=JDBC. This is useful for debugging configuration problems.
- The final load example includes an example of creating a table that is backed by a Parquet file. This is something that you can't do in other databases.



EXERCISE 5

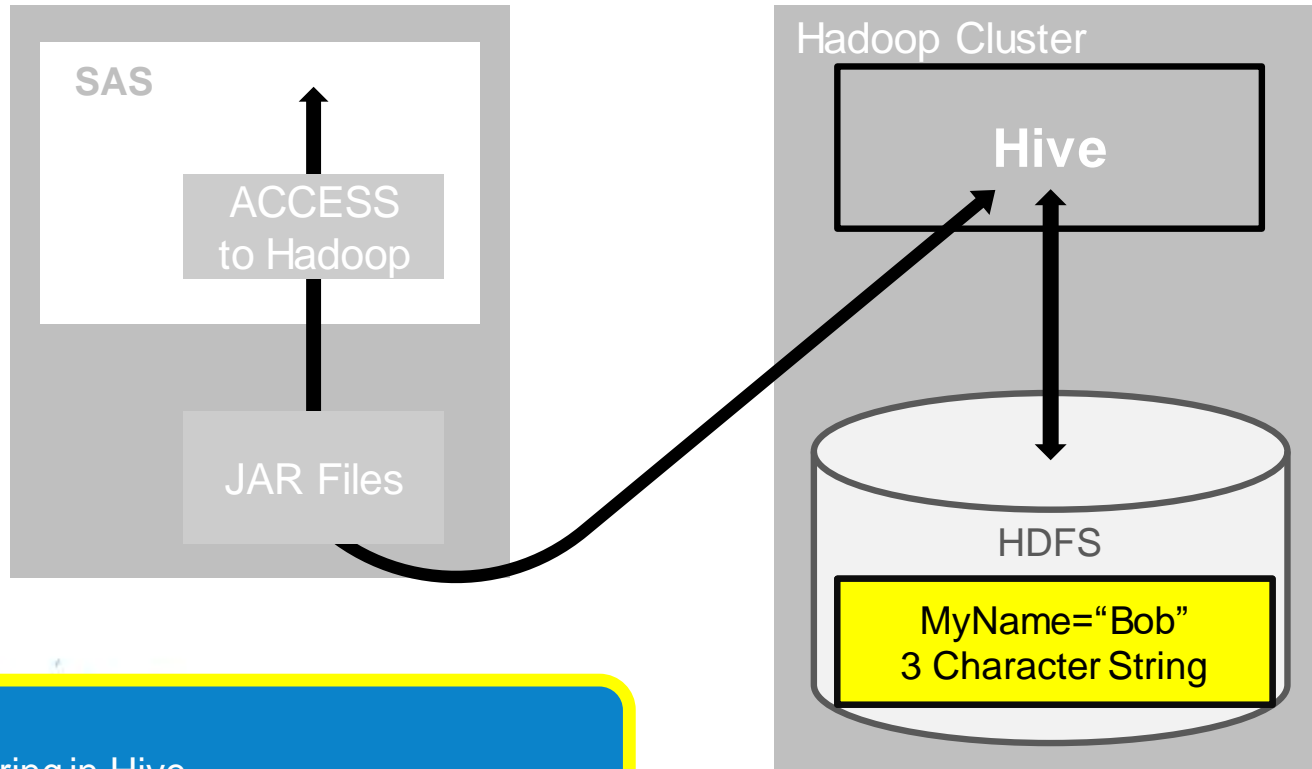
32K STRING THING

Reading Java Strings from Hive



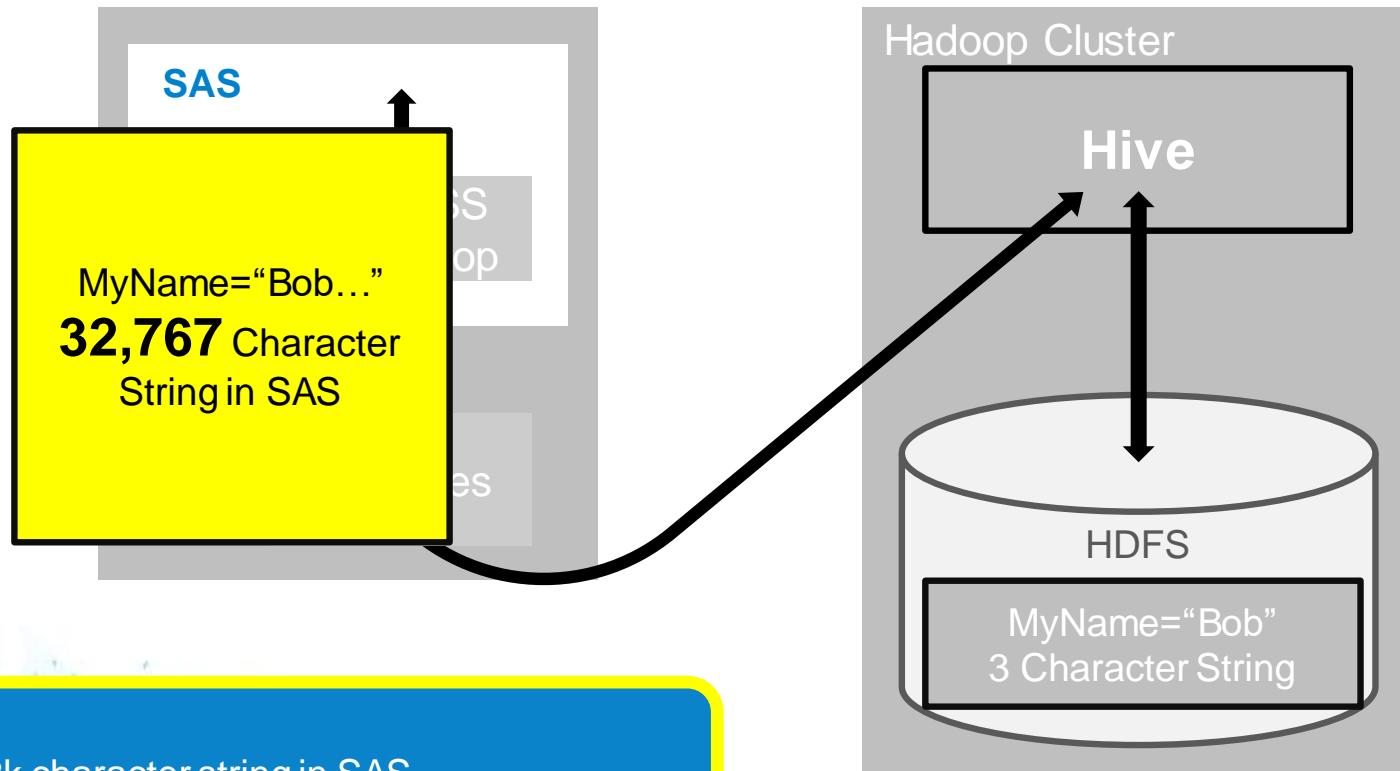
- Java String in Hive

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

Results Viewer - SAS Output														
Location:	hdfs://sasserver.demo.sas.com:8020/user/sasxjb/letters													
Table Type:	EXTERNAL_TABLE													
Table Parameters:	<table><tr><th>Column Name</th><th>Type</th></tr><tr><td>EXTERNAL</td><td>TRUE</td></tr><tr><td>SASFMT:single_character</td><td>CHAR(1)</td></tr><tr><td>last_modified_by</td><td>sasxjb</td></tr><tr><td>last_modified_time</td><td>1404850780</td></tr><tr><td>transient_lastDdlTime</td><td>1404850780</td></tr></table>		Column Name	Type	EXTERNAL	TRUE	SASFMT:single_character	CHAR(1)	last_modified_by	sasxjb	last_modified_time	1404850780	transient_lastDdlTime	1404850780
Column Name	Type													
EXTERNAL	TRUE													
SASFMT:single_character	CHAR(1)													
last_modified_by	sasxjb													
last_modified_time	1404850780													
transient_lastDdlTime	1404850780													

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;
```

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

EX 05 – 32K String Thing Insights

- Use VARCHAR() when possible.
- Extended table attributes can help by letting SAS implicit pass-through know to shorten a string column.
- No space errors are a problem that can be caused by this. Remember the tech support story.
- Explicit pass-through does not honor the extended table attributes.



EXERCISE 6

IN-DATABASE PROCEDURES

EX 05 – In-Database Insights

- Used by default.
- Controlled by SQLGENERATION= option. Make sure you haven't turned it off.
- This is your friend:
 - **OPTIONS SASTRACE=',,,d' SASTRACE=saslog NOSTSUFFIX;**



SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.



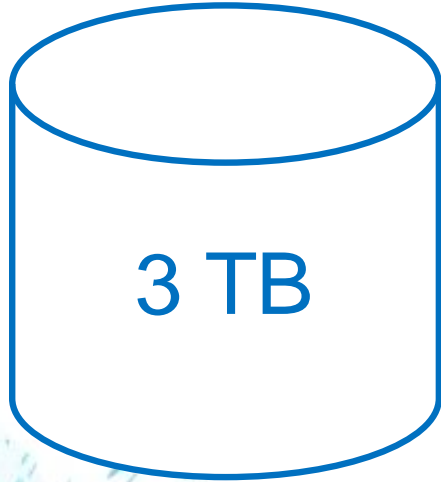
#SASGF





Why Hadoop?

HOW MUCH DOES THIS DRIVE COST?



HOW MUCH DOES THIS DRIVE COST?

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



3 TB

1980

\$1,312,500,000

HOW MUCH DOES THIS DRIVE COST?

That's \$0.03 per GB!

3 TB

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

HOW MUCH DOES THIS DRIVE COST?

That's \$0.03 per GB!

TODAY

\$92

2010

\$270

2005

\$3,720

Insight: Disk Space is **FREE!**

1980

\$1,312,500,000

IT'S NOT JUST ABOUT COST!

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



3 TB

IT'S NOT JUST ABOUT COST!

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

3 TB

4.17 Hours

IT'S NOT JUST ABOUT COST!

How long does it
take to add 10 TB?

What happens if you add **more disks**?

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

1 disk

4.17 hr

100 disks

2.5 min

1000 disks

15 sec

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

1 disk

4.17 hr

Insight: More Disks are **FASTER!**

What is Hadoop?

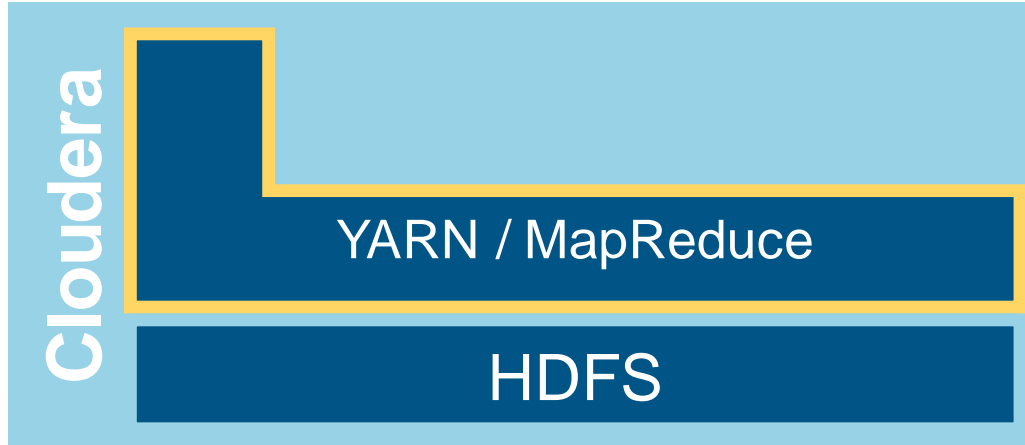
HADOOP IS A STORAGE PLATFORM

Cloudera

HDFS

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

HADOOP IS A PROCESSING PLATFORM



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

HADOOP IS A PROCESSING PLATFORM

MapReduce has 3 phases:

Map Phase

- Reads Data
- Key/Value Pairs

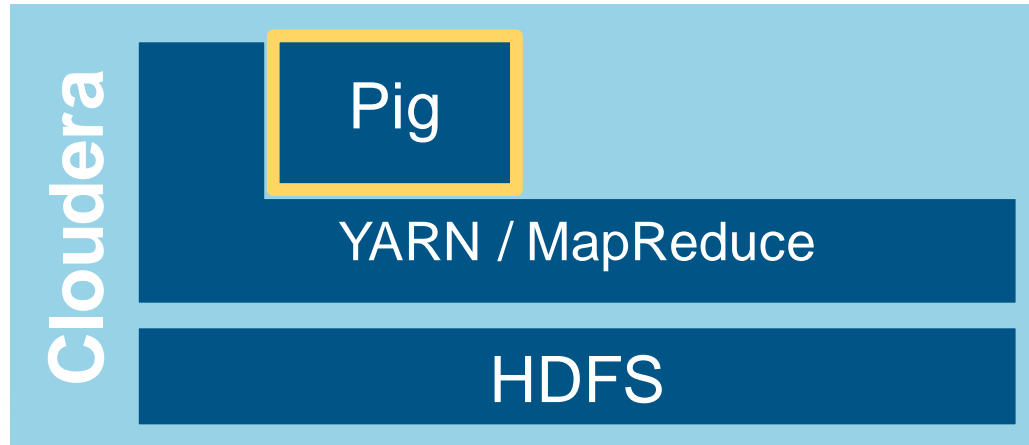
Shuffle + Sort

- Moves data between nodes to prepare for the reduce phase

Reduce Phase

- Combines intermediate keys
- Summary/Aggregation

Apache Pig



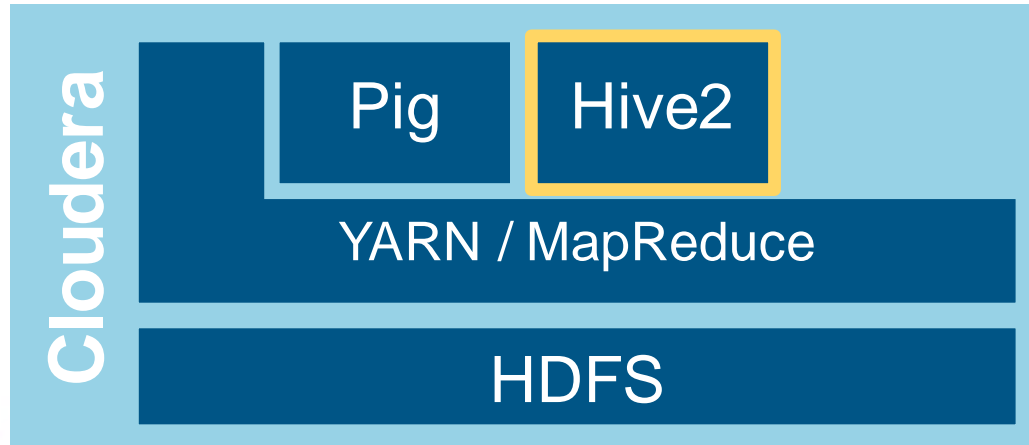
- Scripting Language
- Higher level than programming Java MapReduce
- Pig Latin scripts are converted to MapReduce jobs
- Great for joining data
- Great for transforming data

Apache Pig: Example Program

- Distributed Processing

```
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

Cloudera

- SQL on Hadoop
- Similar to traditional

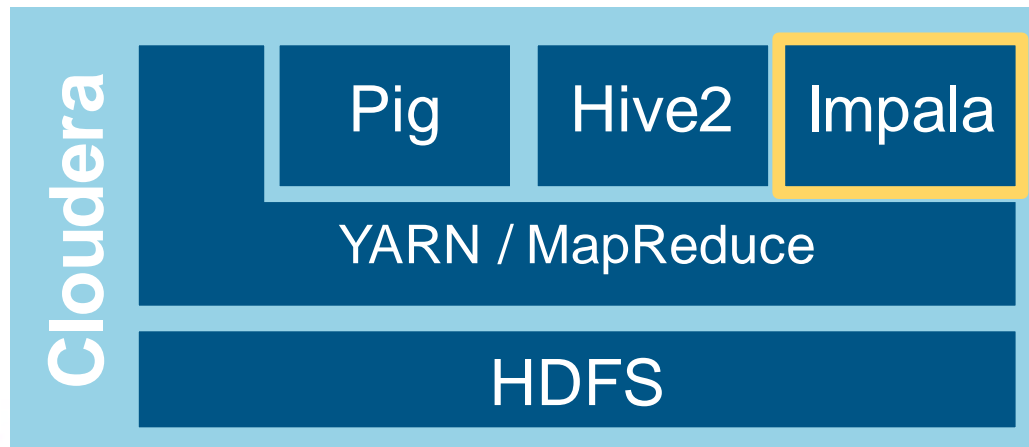
```
SELECT zipcode, SUM(cost) AS total
FROM customers
JOIN orders
ON (customers.cust_id = orders.cust_id)
WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```

underlying file format

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