Module 6

Understanding Impala

Thanachart Numnonda, Executive Director, IMC Institute

Thanisa Numnonda, Faculty of Information Technology,

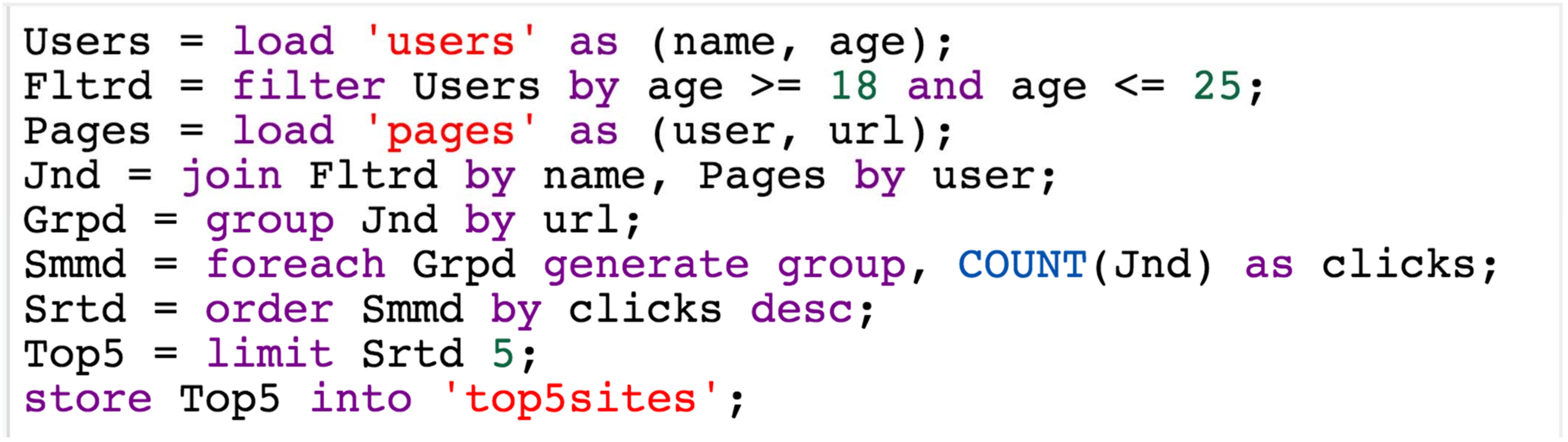
King Mongkut's Institute of Technology Ladkrabang

Introduction

A high-level platform for creating MapReduce programs Using Hadoop

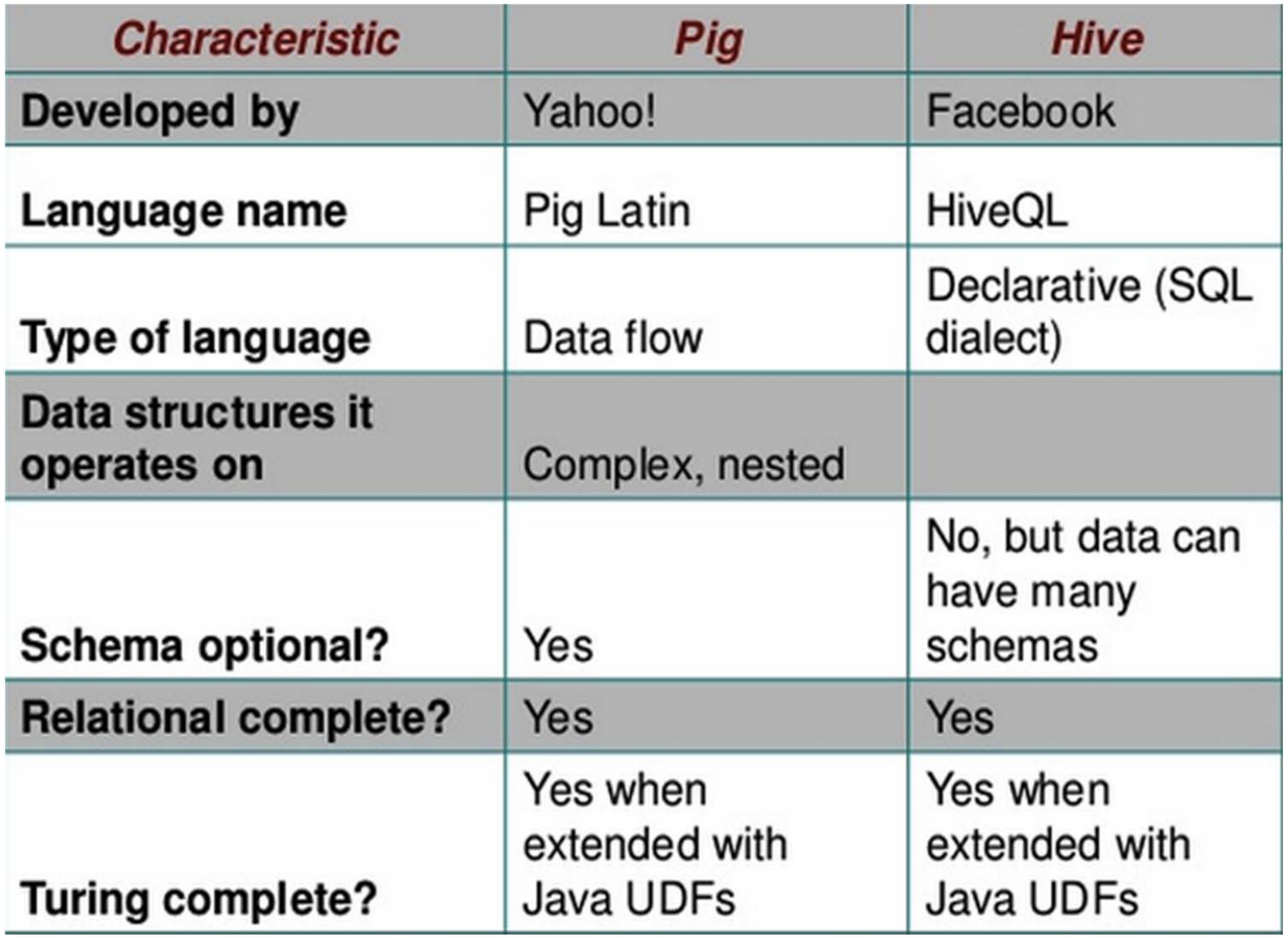
Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

Pig Latin



```
Users = load 'users' as (name, age);
Fltrd = filter Users by age >= 18 and age <= 25;
Pages = load 'pages' as (user, url);
Jnd = join Fltrd by name, Pages by user;
Grpd = group Jnd by url;
Smmd = foreach Grpd generate group, COUNT(Jnd) as clicks;
Srtd = order Smmd by clicks desc;
Top5 = limit Srtd 5;
store Top5 into 'top5sites';
```

Pig v.s. Hive



Characteristic	*Pig*	*Hive*
Developed by	Yahoo!	Facebook
Language name	Pig Latin	HiveQL
Type of language	Data flow	Declarative (SQL dialect)
Data structures it operates on	Complex, nested	
Schema optional?	Yes	No, but data can have many schemas
Relational complete?	Yes	Yes
Turing complete?	Yes when extended with Java UDFs	Yes when extended with Java UDFs

Hive.apache.org

Hive (Revised)

Data Set from GroupLens with

- ratings 27,753,444
- user 280,000
- movie 58,098

http://files.grouplens.org/datasets/movielens/ml-latest.zip

Create “ratings” Table

```
CREATE EXTERNAL TABLE ratings (userId bigint,
    movieId smallint, rating float, timestamp int) ROW
    FORMAT DELIMITED FIELDS TERMINATED
    BY ','  location '/user/cloudera/ratings'
    tblproperties("skip.header.line.count"="1");
```

Already have a file “ratings.csv” in /user/cloudera/ratings/

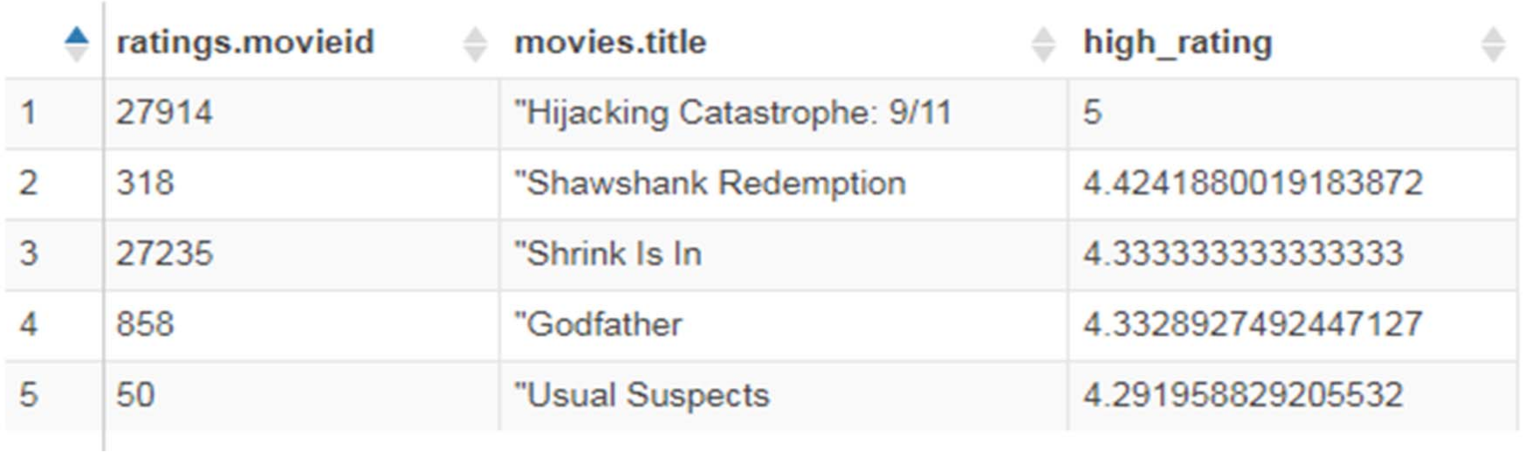
Create “movies” Table

```
CREATE EXTERNAL TABLE movies (movieId int,
    title string, genres string) ROW FORMAT
    DELIMITED FIELDS TERMINATED BY ','
    LOCATION '/user/cloudera/movies'
    tblproperties("skip.header.line.count"="1");
```

Already have a file “movies.csv” in /user/cloudera/movies/

Query Top 5 Rating Movies

```
SELECT ratings.movieId, movies.title, avg(rating) AS
    high_rating FROM ratings
LEFT JOIN movies ON ratings.movieId = movies.movieId
GROUP BY ratings.movieId, movies.title
ORDER BY high_rating desc limit 5;
```



	ratings.movieid	movies.title	high_rating
1	27914	"Hijacking Catastrophe: 9/11	5
2	318	"Shawshank Redemption	4.424188019183872
3	27235	"Shrink Is In	4.333333333333333
4	858	"Godfather	4.332892749244712 7
5	50	"Usual Suspects	4.291958829205532

Drop the Old Table and Create New Table using OpenCSVSerde

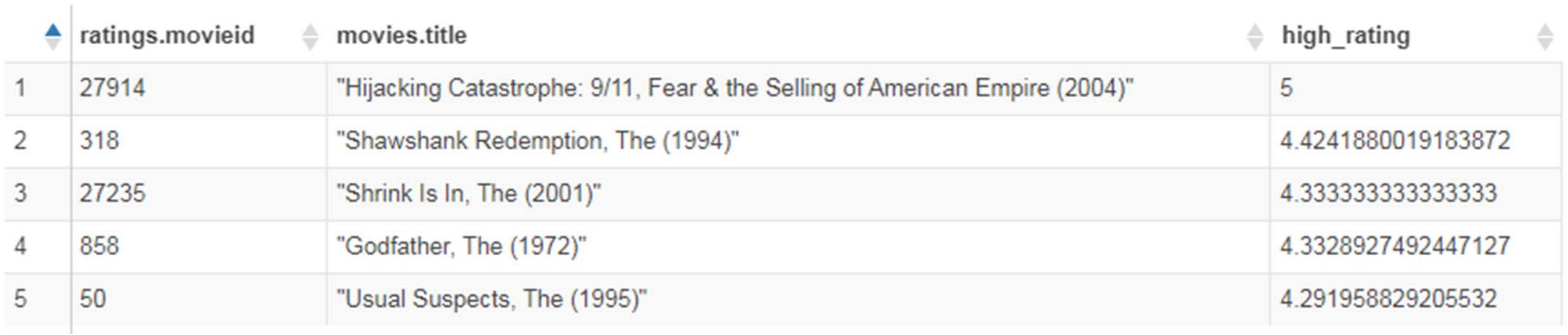
```
CREATE EXTERNAL TABLE movies (movieId int,
   title string, genres string) ROW FORMAT SERDE
   'org.apache.hadoop.hive.serde2.OpenCSVSerde'
   LOCATION '/user/cloudera/movies';
```

Default :

with serdeproperties ("separatorChar"=",", "quoteChar"="\"", "escapeChar"="\\")

New Query

SELECT ratings.movieId, movies.title, avg(rating) AS high_rating FROM ratings LEFT JOIN movies ON ratings.movieId = movies.movieId GROUP BY ratings.movieId, movies.title ORDER BY high_rating desc limit 5;



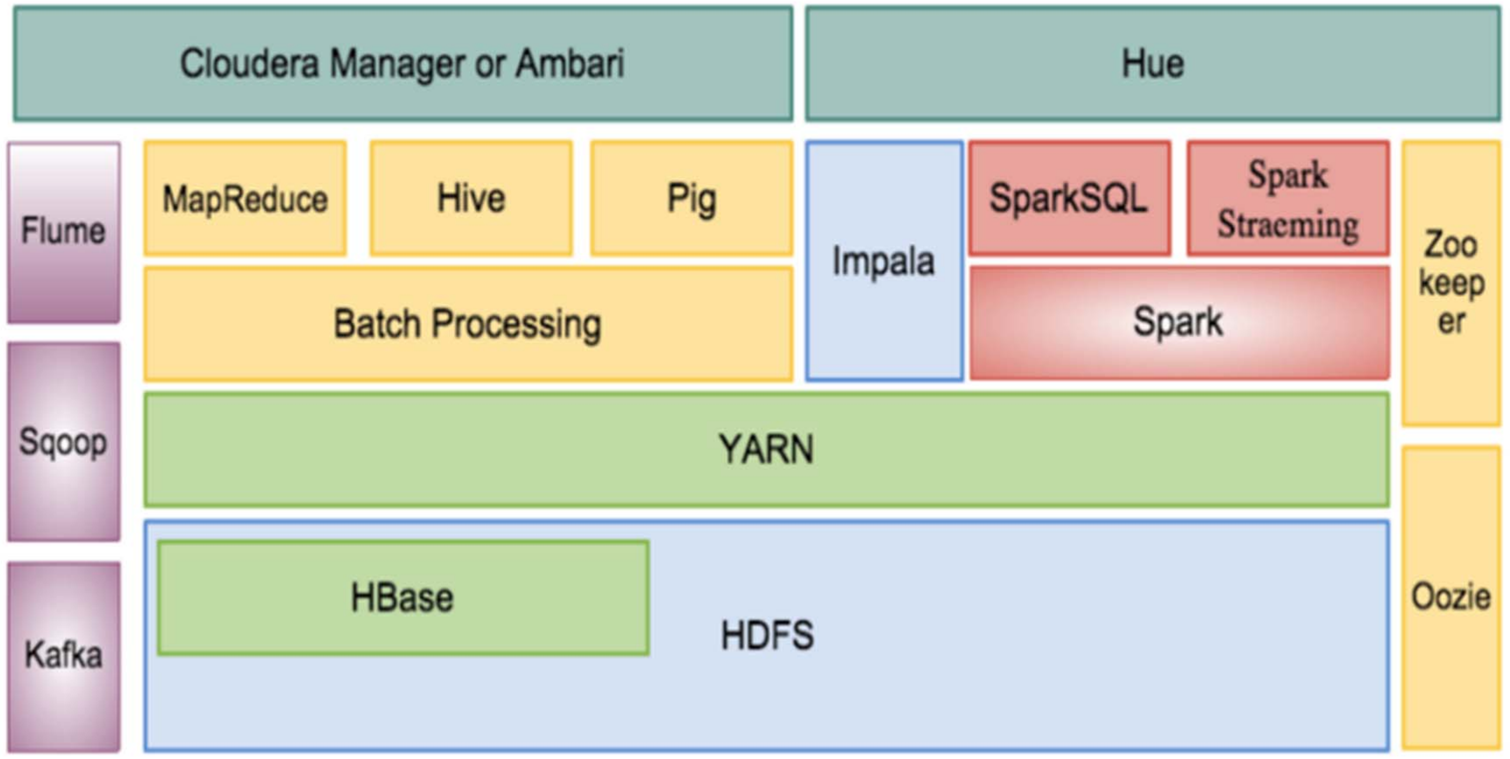
	ratings.movieid	movies.title	high_rating
1	27914	"Hijacking Catastrophe: 9/11, Fear & the Selling of American Empire (2004)"	5
2	318	"Shawshank Redemption, The (1994)"	4.4241880019183872
3	27235	"Shrink Is In, The (2001)"	4.333333333333333
4	858	"Godfather, The (1972)"	4.3328927492447127
5	50	"Usual Suspects, The (1995)"	4.291958829205532

Impala

Why do we need it?

SPEED

Hadoop Ecosystem



Support queries takes from milliseconds to hours (near real-time)

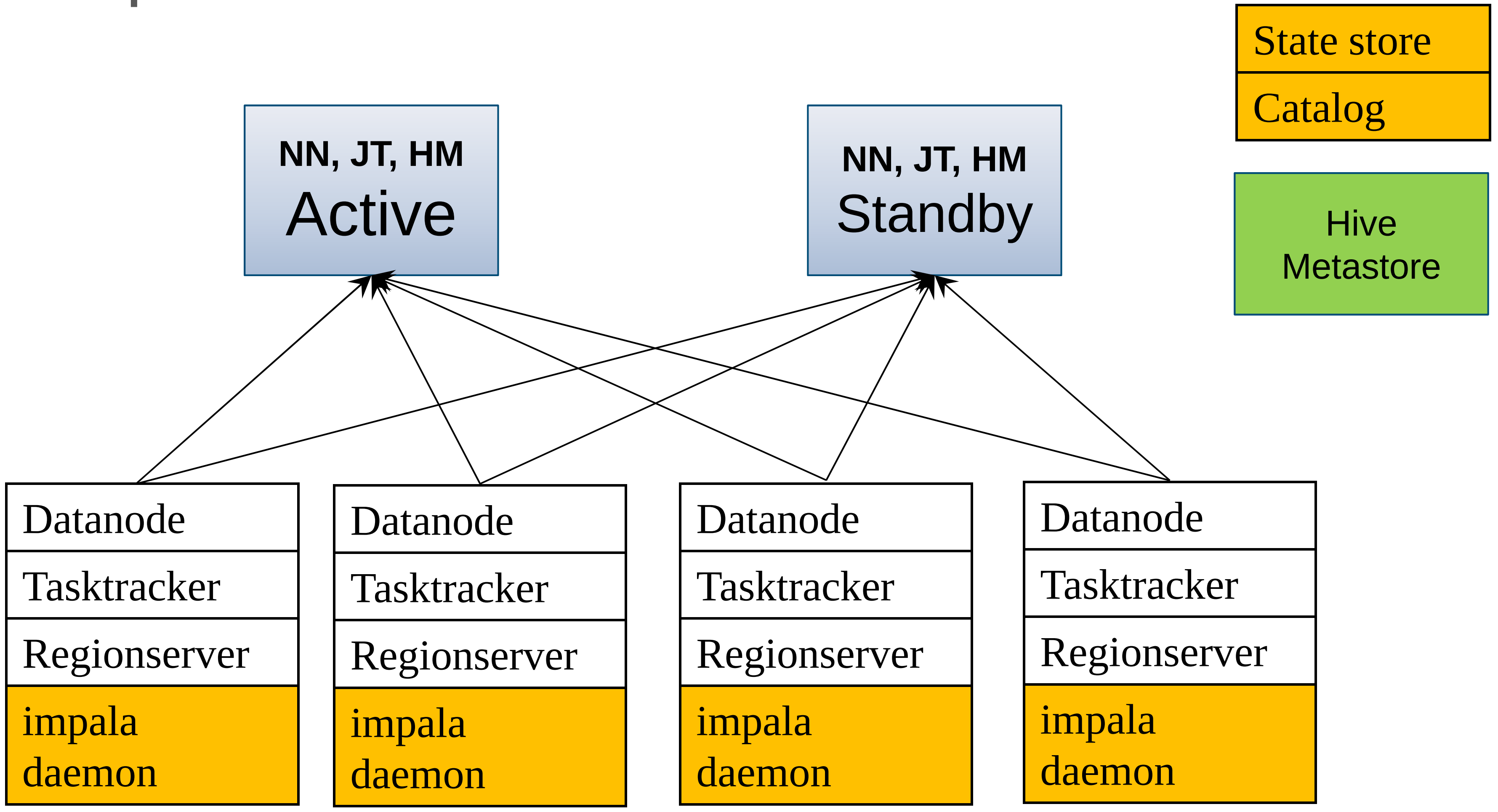
About Impala

- Developed by Cloudera
- Open source under Apache License
- Current version is 3.1.0
- Connect via ODBC/JDBC/hue/impala-shell

Benefits

- High Performance
  - C++
  - Direct access to data (No JVM, No MapReduce).
  - In-memory query execution
- Flexibility
  - Query across existing data (No Duplication)
  - Support multiple Hadoop file format
- Scalable
  - Scale out by adding nodes

Impala Architecture



Components

- Impala daemon
  - Runs on every node.
  - Collocate with data nodes.
  - Handle client requests related to query execution.
    - User can submit request to impala daemon running on any node and that node serve as coordinator node
  - Handle query planning & execution.

Components (Cont.)

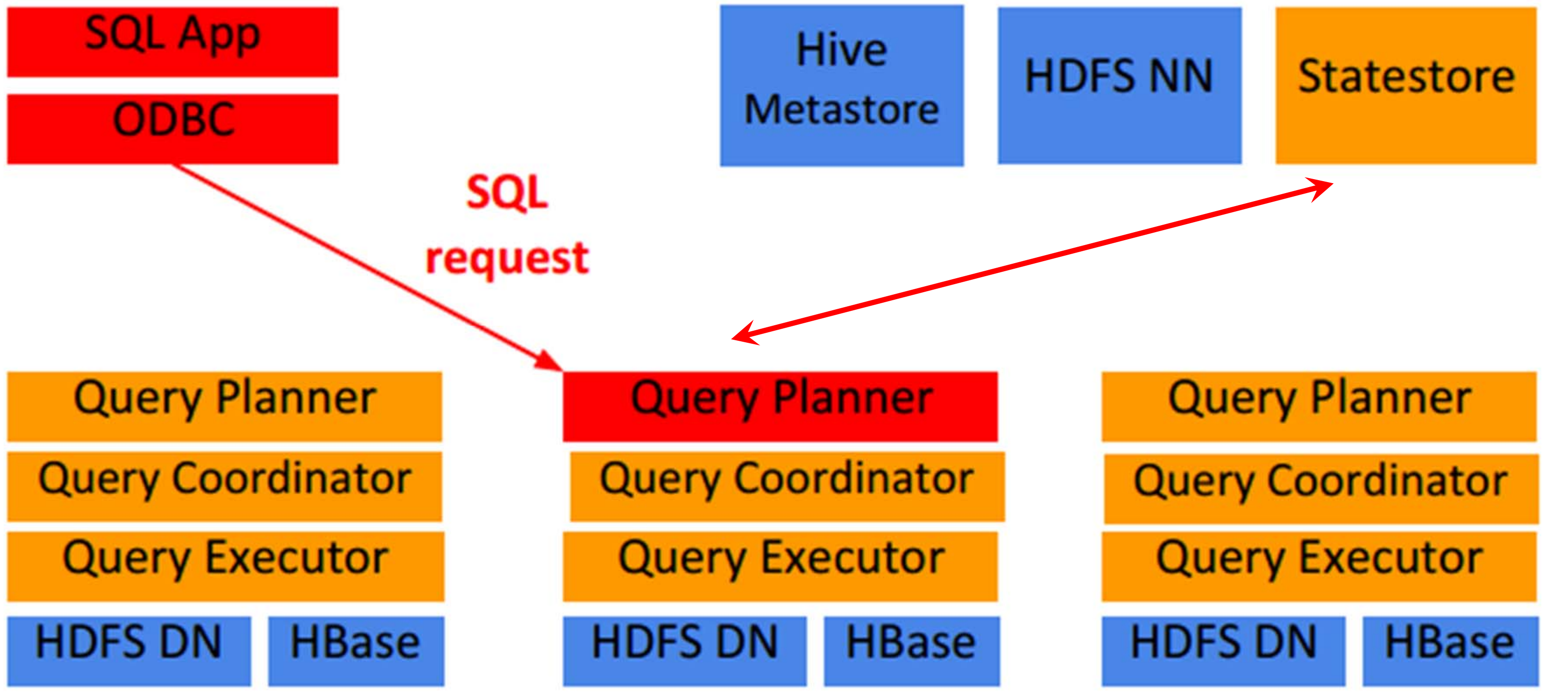
- State store daemon
  - Provides name service
  - Metadata distribution
    - Used for finding data.
  - Communicates with impala daemons to confirm which node is healthy and can accept new work
- Catalog daemon
  - broadcast metadata changes from impala SQL statements to all the impala daemons
    - via the state store daemon

Fault tolerance

- No fault tolerance for impala daemons
  - A node failed, the query failed
- State-store offline
  - query execution still function normally
  - can not update metadata(create, alter…)
  - if another impala daemon goes down, then entire cluster can not execute any query
- Catalog offline
  - can not update metadata

Impala Architecture: Query Execution

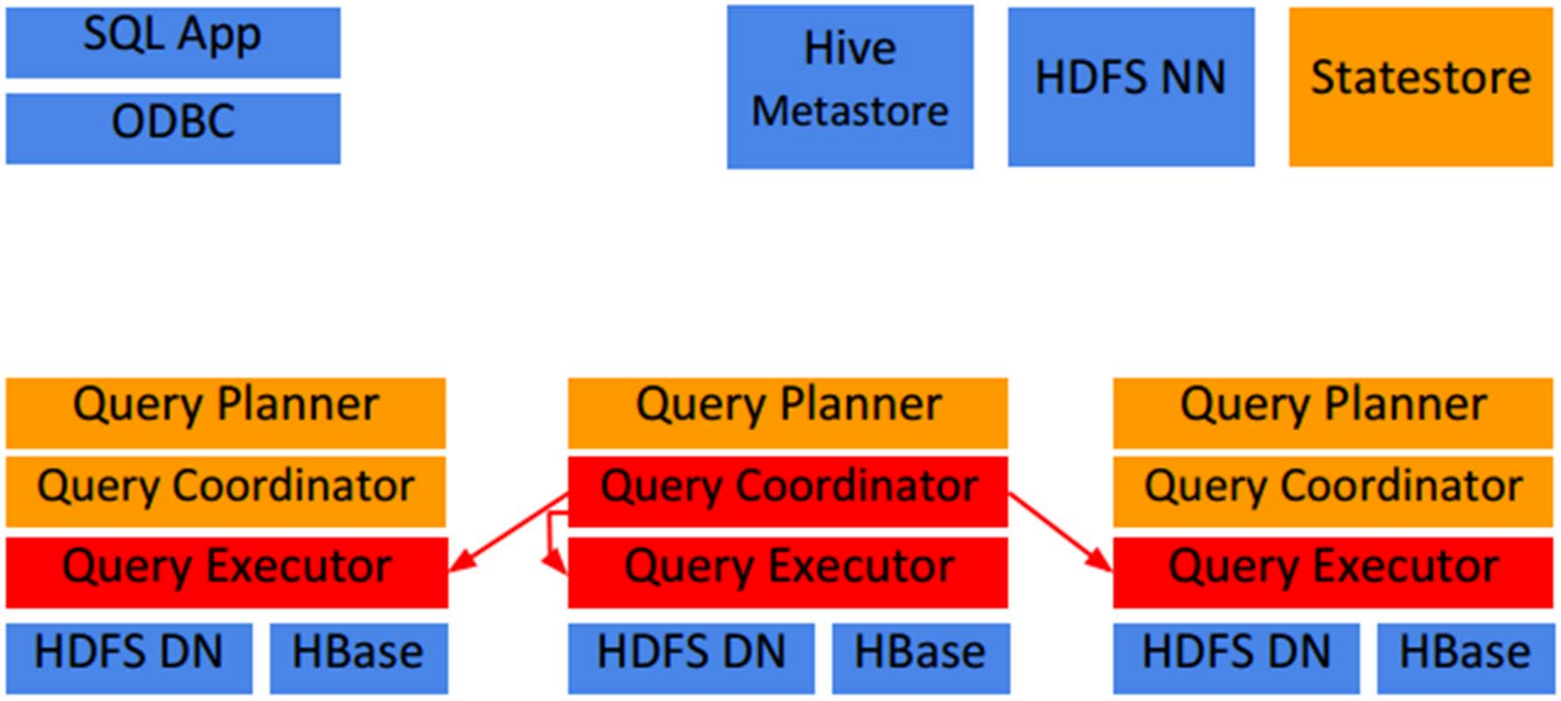
Request arrives via odbc/jdbc



Impala Architecture: Query Execution

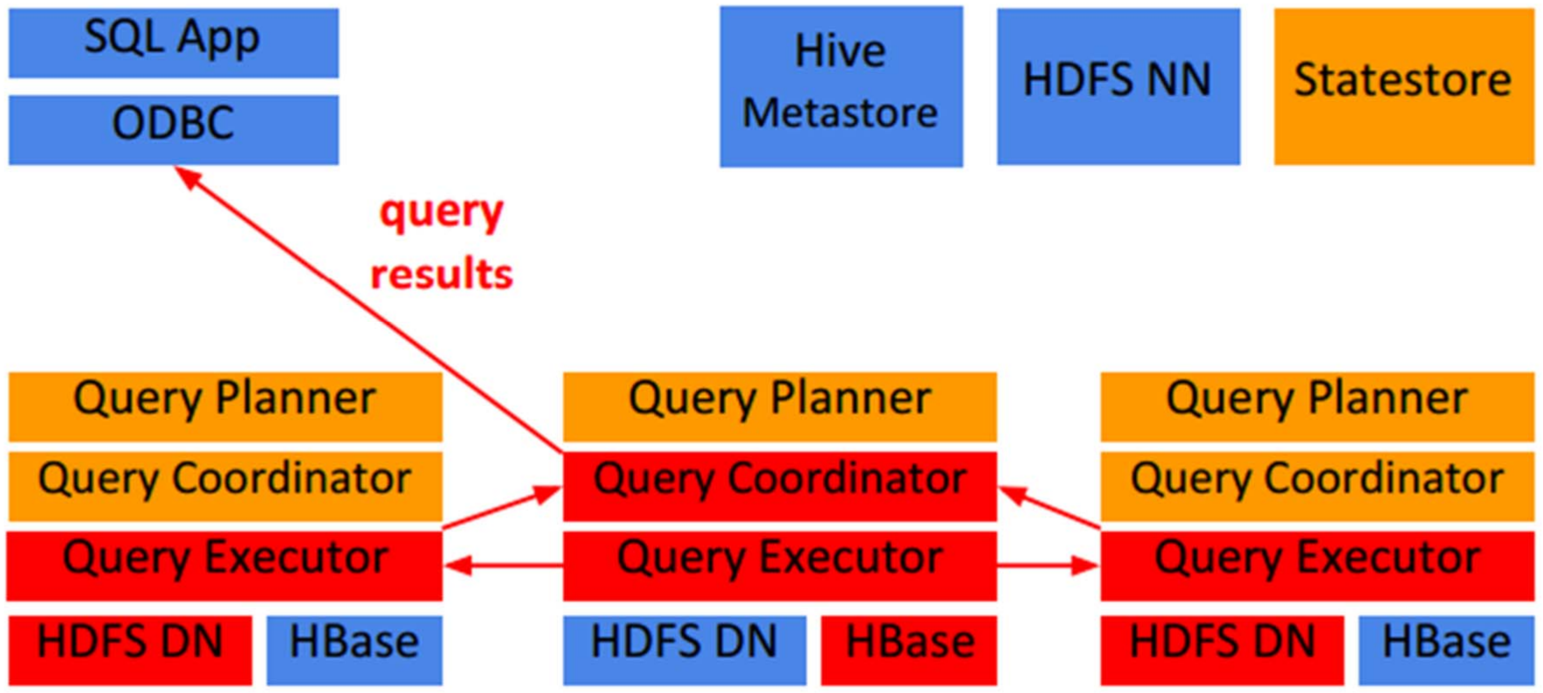
Planner turns request into collections of plan fragments

Coordinator initiates execution on remote impalad's

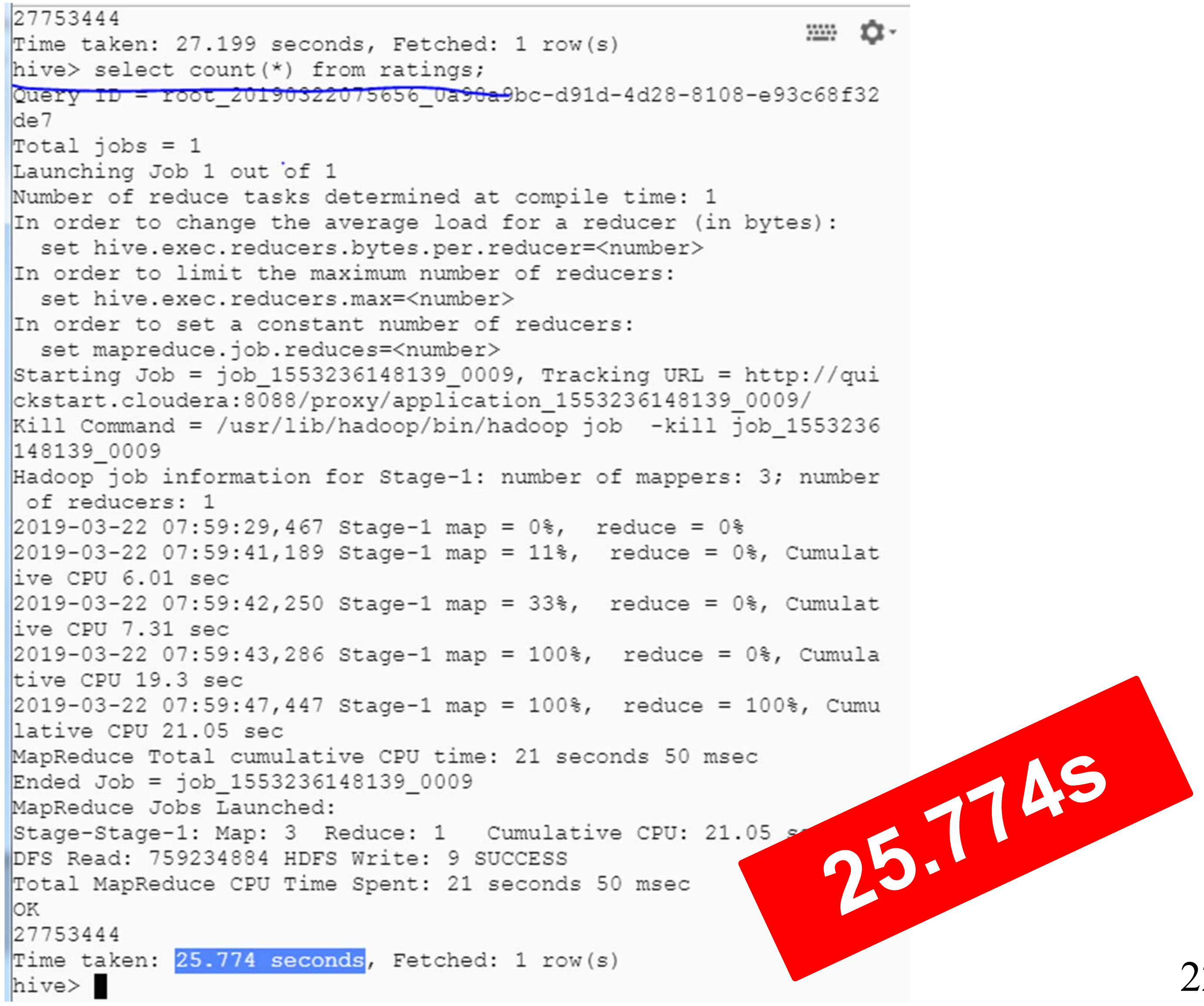


Impala Architecture: Query Execution

Intermediate results are streamed between impalad's Query results are streamed back to client

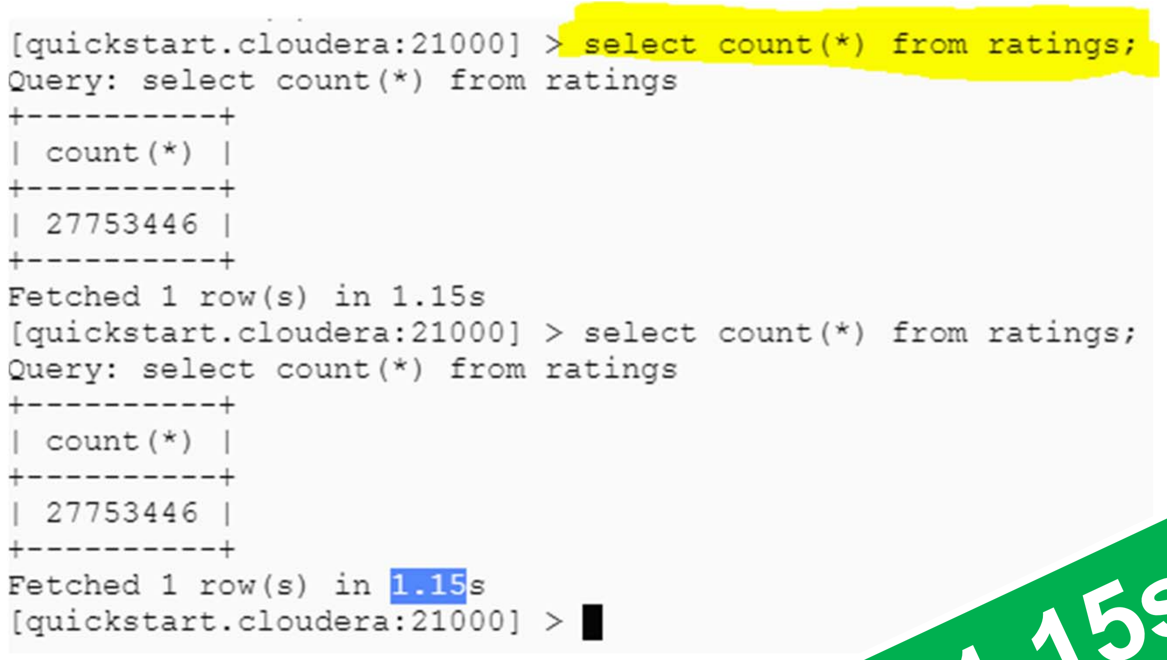


Usage Time in Hive



```
27753444
Time taken: 27.199 seconds, Fetched: 1 row(s)
hive> select count(*) from ratings;
Query ID = root_20190322075656_0a98a9bc-d91d-4d28-8108-e93c68f32
de7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1553236148139_0009, Tracking URL = http://qui
ckstart.cloudera:8088/proxy/application_1553236148139_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1553236
148139_0009
Hadoop job information for Stage-1: number of mappers: 3; number
 of reducers: 1
2019-03-22 07:59:29,467 Stage-1 map = 0%,  reduce = 0%
2019-03-22 07:59:41,189 Stage-1 map = 11%,  reduce = 0%, Cumulat
ive CPU 6.01 sec
2019-03-22 07:59:42,250 Stage-1 map = 33%,  reduce = 0%, Cumulat
ive CPU 7.31 sec
2019-03-22 07:59:43,286 Stage-1 map = 100%,  reduce = 0%, Cumula
tive CPU 19.3 sec
2019-03-22 07:59:47,447 Stage-1 map = 100%,  reduce = 100%, Cumu
lative CPU 21.05 sec
MapReduce Total cumulative CPU time: 21 seconds 50 msec
Ended Job = job_1553236148139_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 1   Cumulative CPU: 21.05 s
DFS Read: 759234884 HDFS Write: 9 SUCCESS
Total MapReduce CPU Time Spent: 21 seconds 50 msec
OK
27753444
Time taken: 25.774 seconds, Fetched: 1 row(s)
hive>
```

Usage Time in Impala



```
[quickstart.cloudera:21000] > select count(*) from ratings;
Query: select count(*) from ratings
+----------+
| count(*) |
+----------+
| 27753446 |
+----------+
Fetched 1 row(s) in 1.15s
[quickstart.cloudera:21000] > select count(*) from ratings;
Query: select count(*) from ratings
+----------+
| count(*) |
+----------+
| 27753446 |
+----------+
Fetched 1 row(s) in 1.15s
[quickstart.cloudera:21000] >
```

1.15s