COMP810 Data Warehousing and Big Data

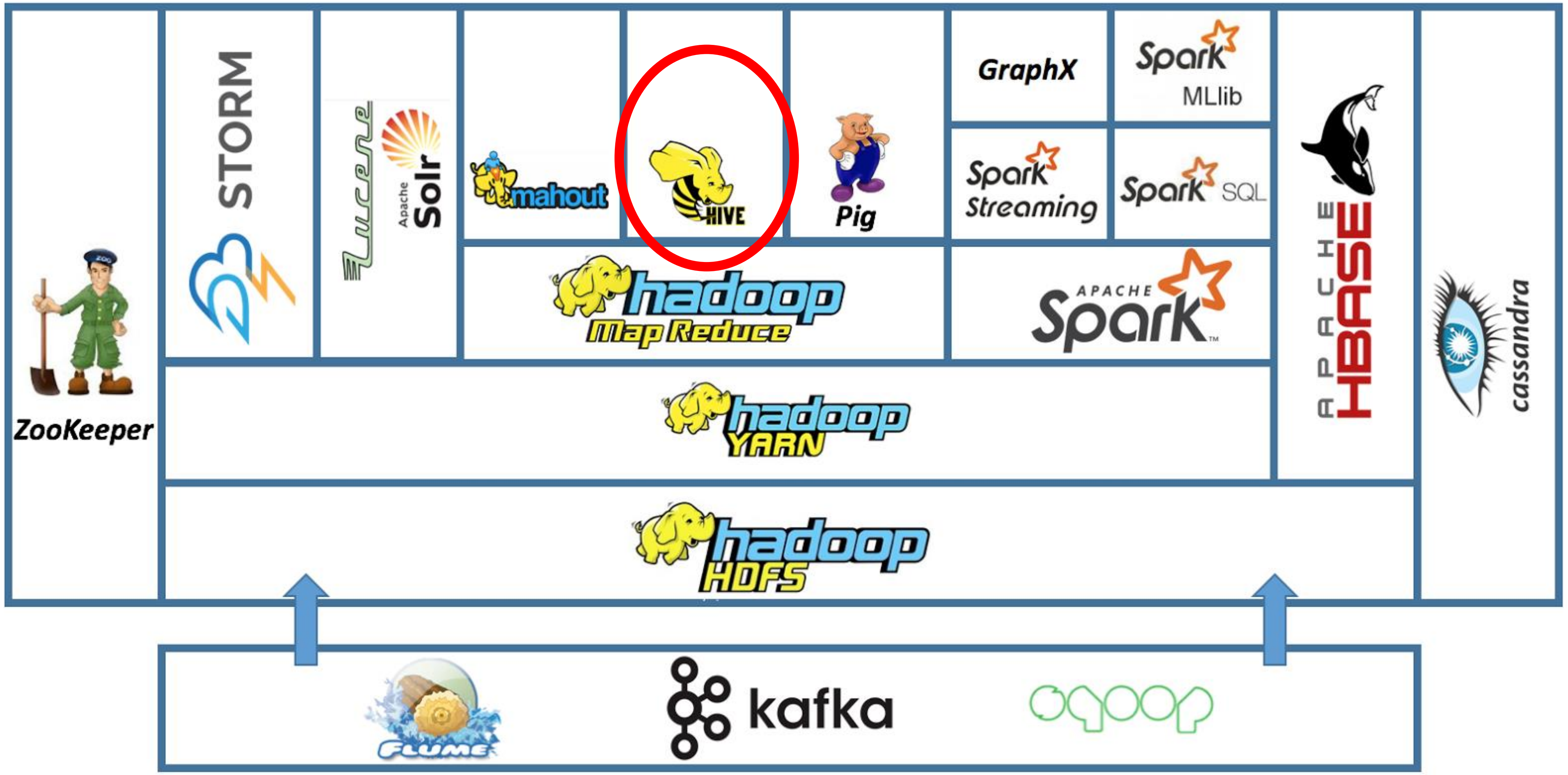
Big Data with Apache Hive

Dr Weihua Li

Outline

- Introduction to Hive
- Hive Architecture
- Hive Query Language (HQL)
- HQL to MapReduce
- Hive Client

Apache Ecosystem



Why Another Data Warehousing System

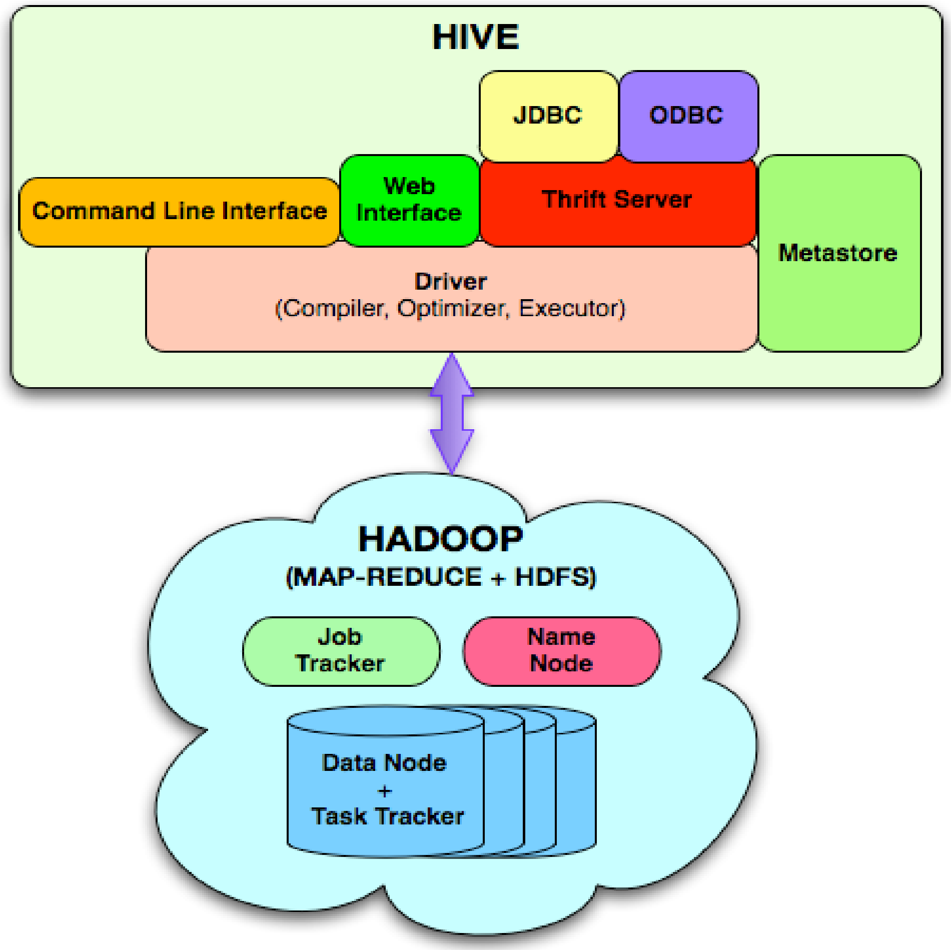
- **Problem**: Data and more data. The statistics in 2018, ***every minute***:
  - Users watch 4,146,600 YouTube videos
  - 456,000 tweets are sent on Twitter
  - Instagram users post 46,740 photos
  - 510,000 comments posted and 293,000 statuses updated
- **Solution**: MapReduce is scalable which is great!

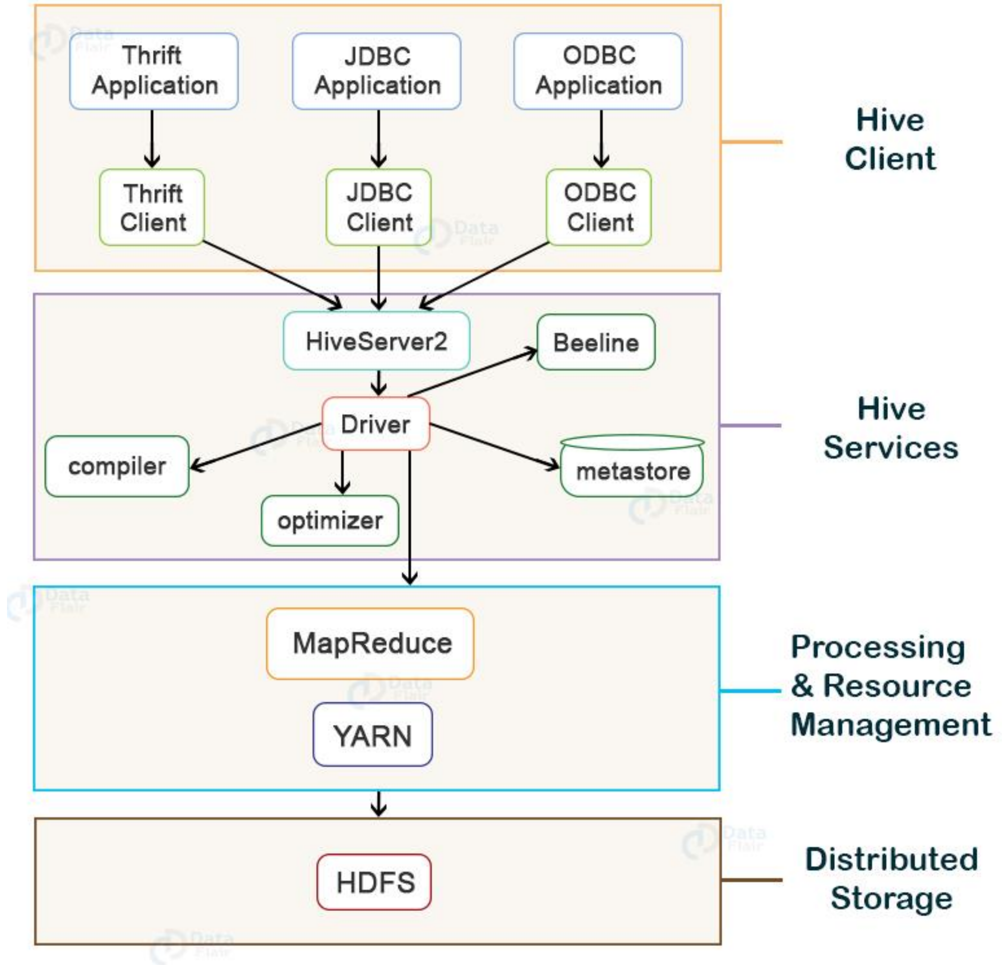
- **Problem**: Not everyone is a MapReduce expert.
  - Most developers are familiar with SQL
  - SQL is easy to code
- **Solution**: Combine SQL with MapReduce
  - Hive on top of Hadoop (open source)

What is Apache Hive

- A database/data warehouse (system) built on top of Hadoop
  - Developed at Facebook
  - Apache open-source project
  - Query and manage structured BIG data
  - SQL-like query language (HiveQL)
  - Rich data types (structs, lists and maps)
  - Efficient implementations of SQL filters, joins and group by on top of map reduce

Hive Architecture and Components

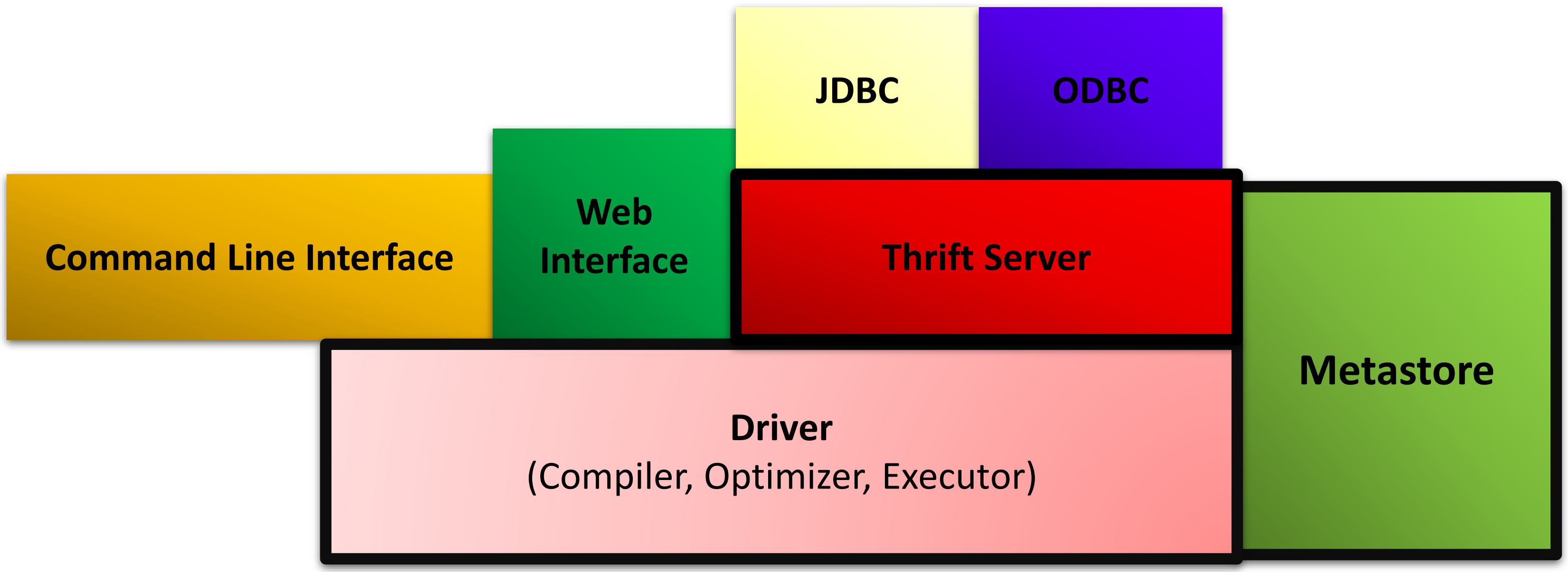




Hive Architecture & Its Components

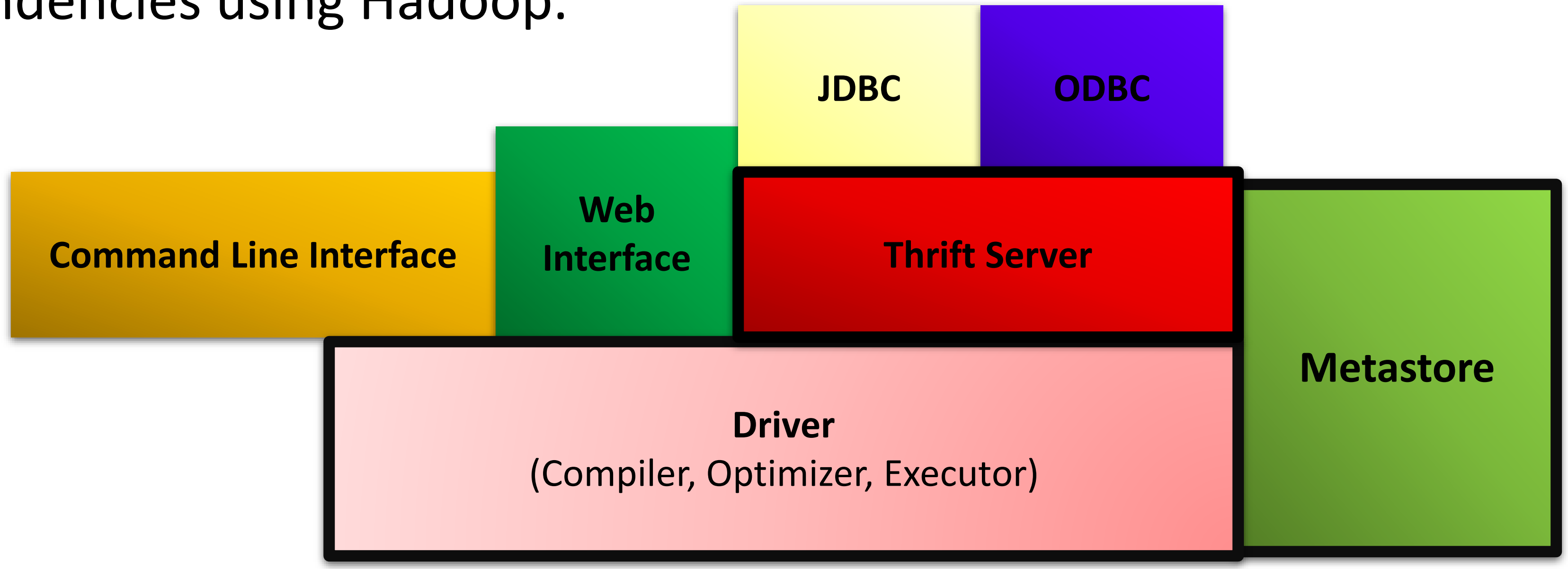
Hive Architecture – Hive Services (1)

- **Metastore**: A small database, stored the system catalogue and metadata about tables, columns, partitions etc.
- **Thrift Server**: Handle the cross-platform communication with Hive, which provide a way of integrating Hive with other applications.



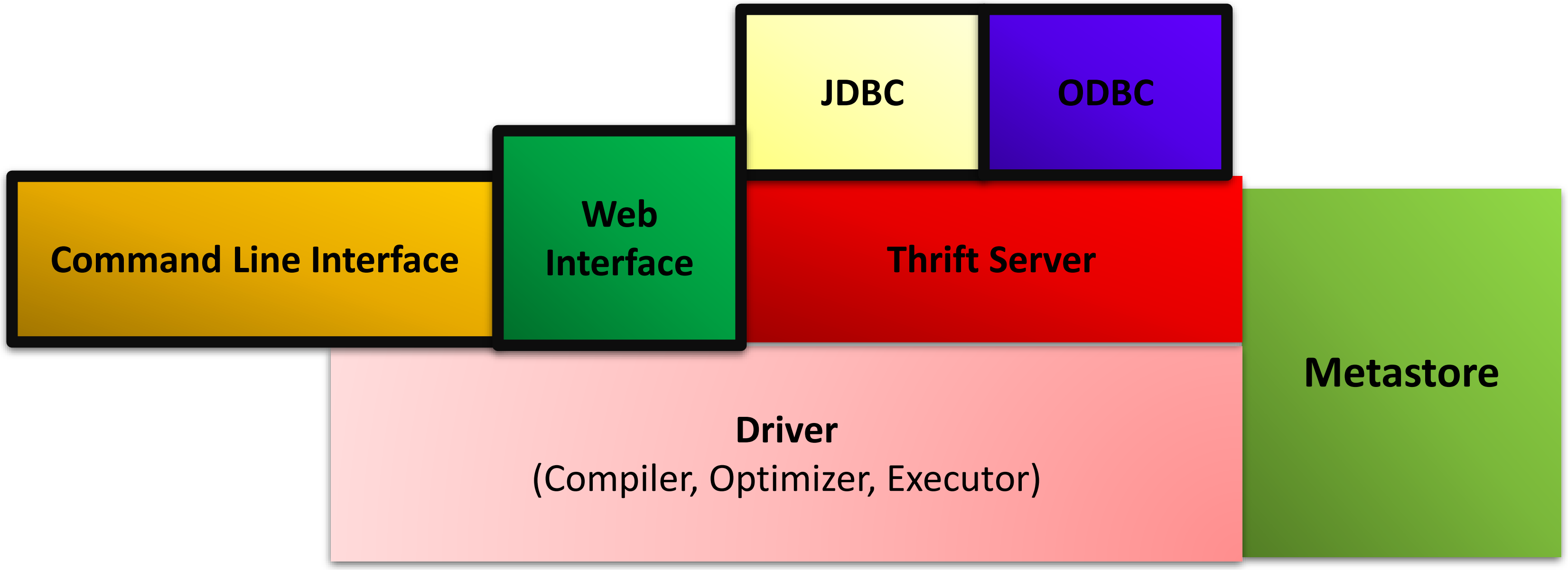
Hive Architecture – Hive Services (2)

- **Driver**: Manage the lifecycle of a HiveQL statement
  - **Compiler**: compile HiveQL into MapReduce tasks.
  - **Optimizer**: Perform the transformation operations on the execution plan and splits the task to improve efficiency and scalability
  - **Executor**: Execute the execution plan created by the compiler in order of their dependencies using Hadoop.

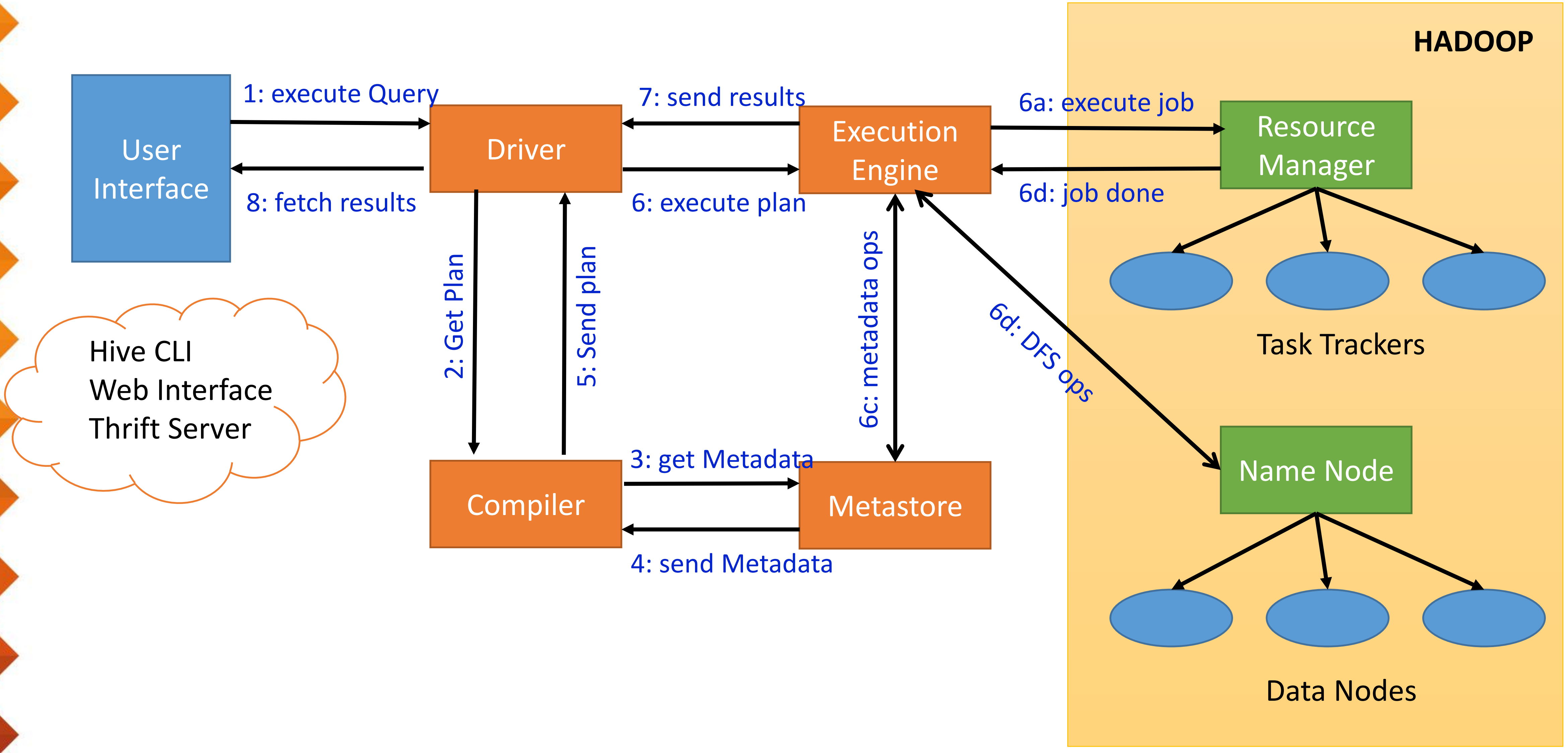


Hive Architecture – Hive Client

- **Client Components**
  - Client component like Command Line Interface(CLI), the web UI and JDBC/ODBC driver.

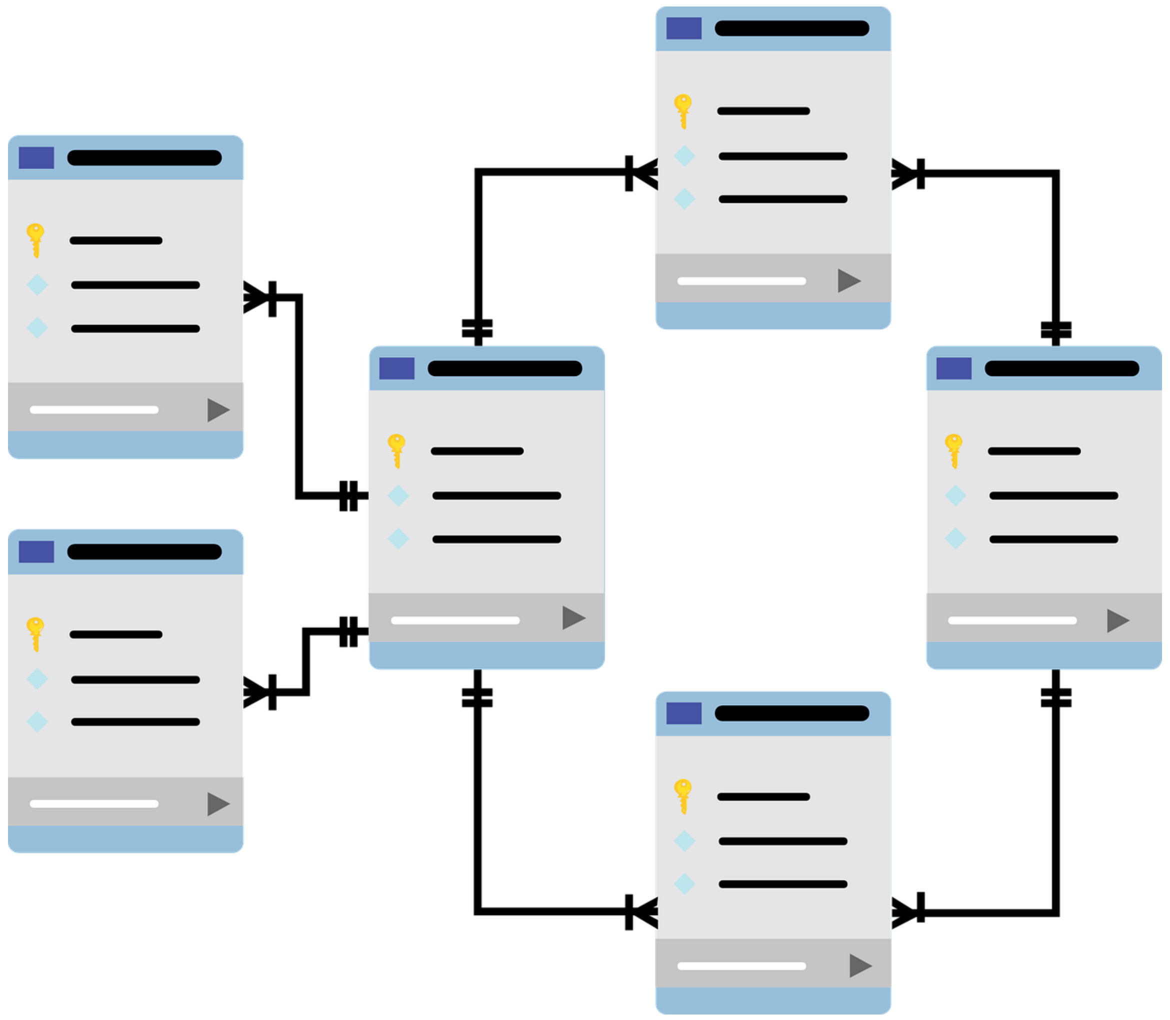


Hive Architecture – Process



Hive Query Language

(HQL)



Hive Data Structure

- Hive organizes data into a set of data structures that facilitate efficient storage, querying, and analysis within the Hadoop ecosystem.
- Store data in a tabular format
- Rows
- Columns
- Leverage partitions
- Data Types: int, float, double, string, boolean, date, etc.

Creating a Hive Table

```
CREATE TABLE page_views(viewTime INT, userid BIGINT,
                        page_url STRING, referrer_url STRING,
                        ip STRING)
PARTITIONED BY(dt STRING, country STRING)
STORED AS TEXTFILE;
```

- Partitioning breaks table into separate files for each (dt, country) pair

  Ex: /hive/page_view/dt=2020-08-26,country=NZ

  /hive/page_view/dt=2020-08-26,country=AUS

A Simple Query

- Find all page views coming from xyz.com in August 2020:

```
SELECT page_views.*
FROM page_views
WHERE page_views.dt >= '2020-08-01'
AND page_views.dt <= '2020-08-31'
AND page_views.referrer_url like '%xyz.com';
```

- Hive only reads partition `2020-08-01,*` instead of scanning entire table

Aggregate Functions

- Aggregate functions perform calculations on a group of values and return a single value.
- They are typically used to summarize data, such as calculating sums, averages, or counts.
- In HQL, when using aggregate functions such as COUNT, SUM, or AVG, any non-aggregated columns, such as u.gender, must be included in the GROUP BY clause.
- This ensures that the query correctly groups the data by the specified columns before applying the aggregate functions.

```
SELECT u.gender, COUNT(u.id)
FROM user as u
GROUP BY u.gender
```

gender	COUNT
MALE	23,941,451
FEMALE	21,231,314

Joins

- Obtain Data from Multiple Hive Tables
- JOIN is a clause that is used for combining specific fields from two tables by using values common to each one.

page_view

pageid	**userid**	time
1	**111**	9:08:01
2	**111**	9:08:13
1	**222**	9:08:14

X

user

userid	age	gender
111	25	female
222	32	male

=

pv_users

pageid	age
1	25
2	25
1	32

```
INSERT INTO TABLE pv_users
SELECT pv.pageid, u.age
FROM page_view pv
JOIN user u ON (pv.userid = u.userid);
```

```
INSERT INTO TABLE pv_users
SELECT pv.pageid, u.age
FROM page_view pv, user u
WHERE pv.userid = u.userid;
```

Aggregate and Joins

- Count users who visited each page by gender:

```
SELECT pv.page_url, u.gender, COUNT(DISTINCT u.id)
FROM page_views pv JOIN user u ON (pv.userid = u.id)
GROUP BY pv.page_url, u.gender
WHERE pv.dt = '2020-08-26';
```

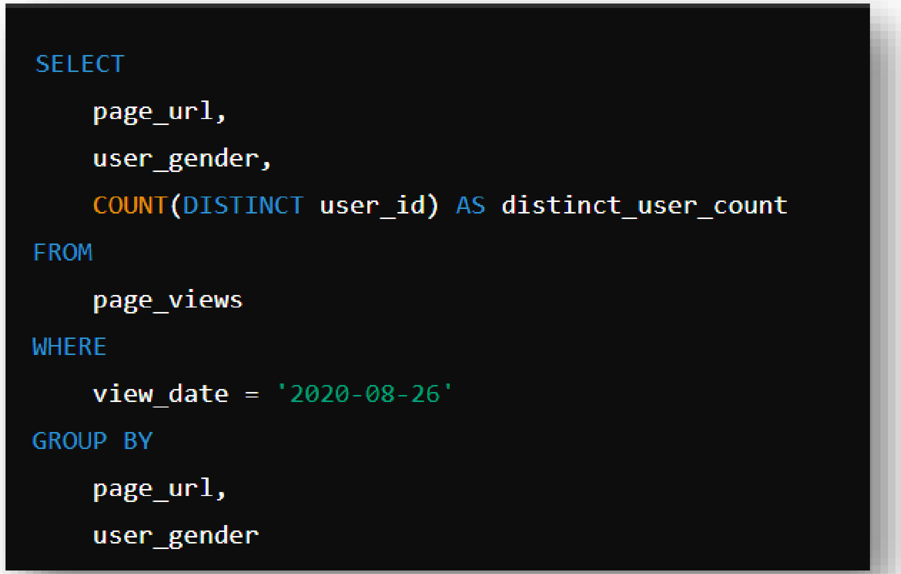
- Sample output:

page_url	gender	count(userid)
home.php	MALE	12,141,412
home.php	FEMALE	15,431,579
photo.php	MALE	23,941,451
photo.php	FEMALE	21,231,314

Generative AI (GenAI) Assistance

- Generate HQL

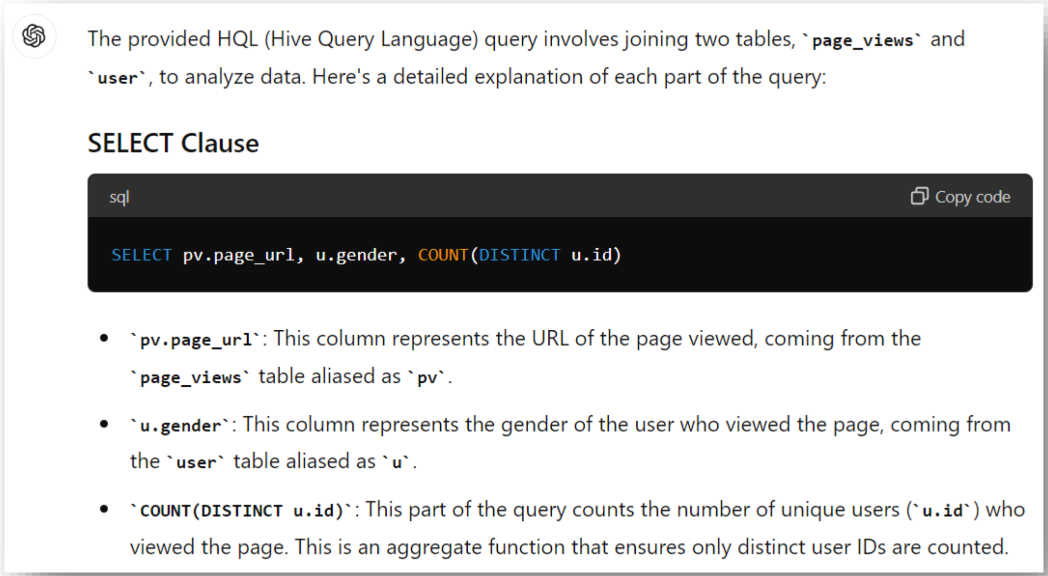
Write an HQL query, which retrieves the URLs of pages viewed, the gender of users, and the count of distinct users who viewed each page on August 26, 2020. The results are grouped by page URL and user gender.



```sql
SELECT
    page_url,
    user_gender,
    COUNT(DISTINCT user_id) AS distinct_user_count
FROM
    page_views
WHERE
    view_date = '2020-08-26'
GROUP BY
    page_url,
    user_gender
```

- Explain HQL

Please explain the HQL:
SELECT pv.page_url, u.gender, COUNT(DISTINCT u.id)
FROM page_views pv JOIN user u ON (pv.userid = u.id)
GROUP BY pv.page_url, u.gender
WHERE pv.dt = '2020-08-26';



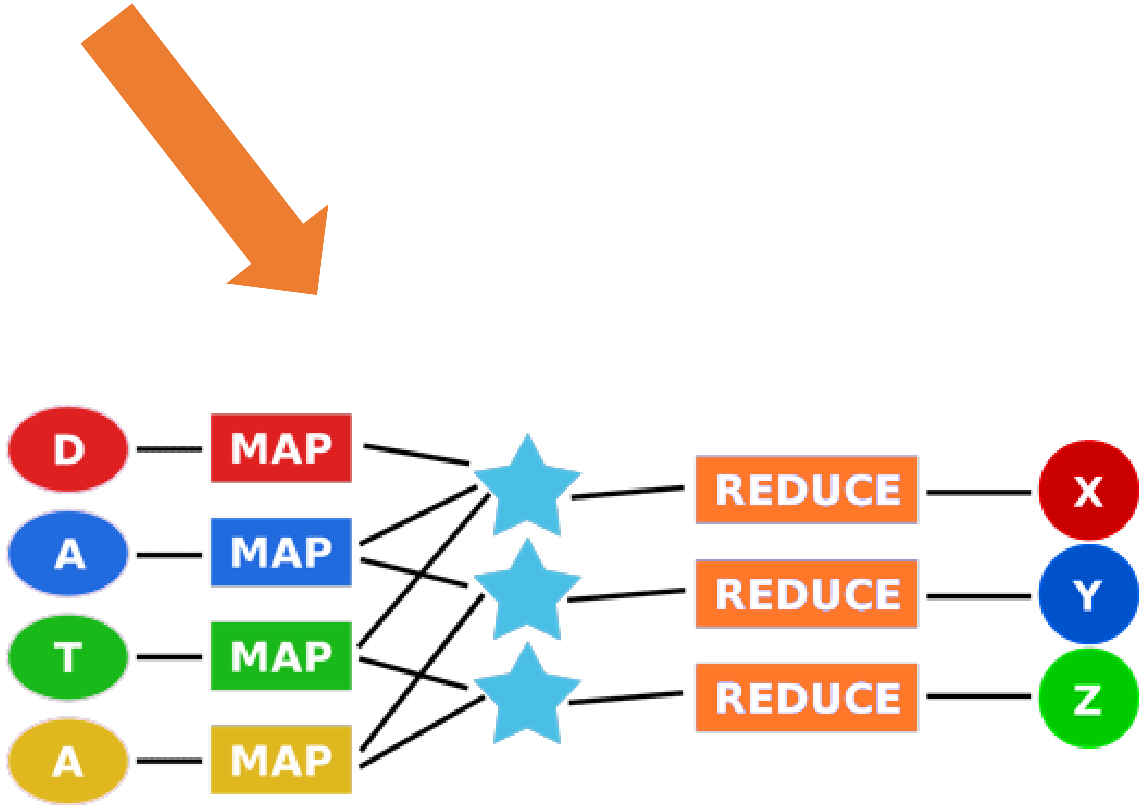
The provided HQL (Hive Query Language) query involves joining two tables, `page_views` and `user`, to analyze data. Here's a detailed explanation of each part of the query:

SELECT Clause

```sql
SELECT pv.page_url, u.gender, COUNT(DISTINCT u.id)
```

- `pv.page_url`: This column represents the URL of the page viewed, coming from the `page_views` table aliased as `pv`.
- `u.gender`: This column represents the gender of the user who viewed the page, coming from the `user` table aliased as `u`.
- `COUNT(DISTINCT u.id)`: This part of the query counts the number of unique users (`u.id`) who viewed the page. This is an aggregate function that ensures only distinct user IDs are counted.

HQL to MapReduce



Hive QL – Join

page_view

pageid	**userid**	time
1	**111**	9:08:01
2	**111**	9:08:13
1	**222**	9:08:14

X

user

userid	age	gender
111	25	female
222	32	male

=

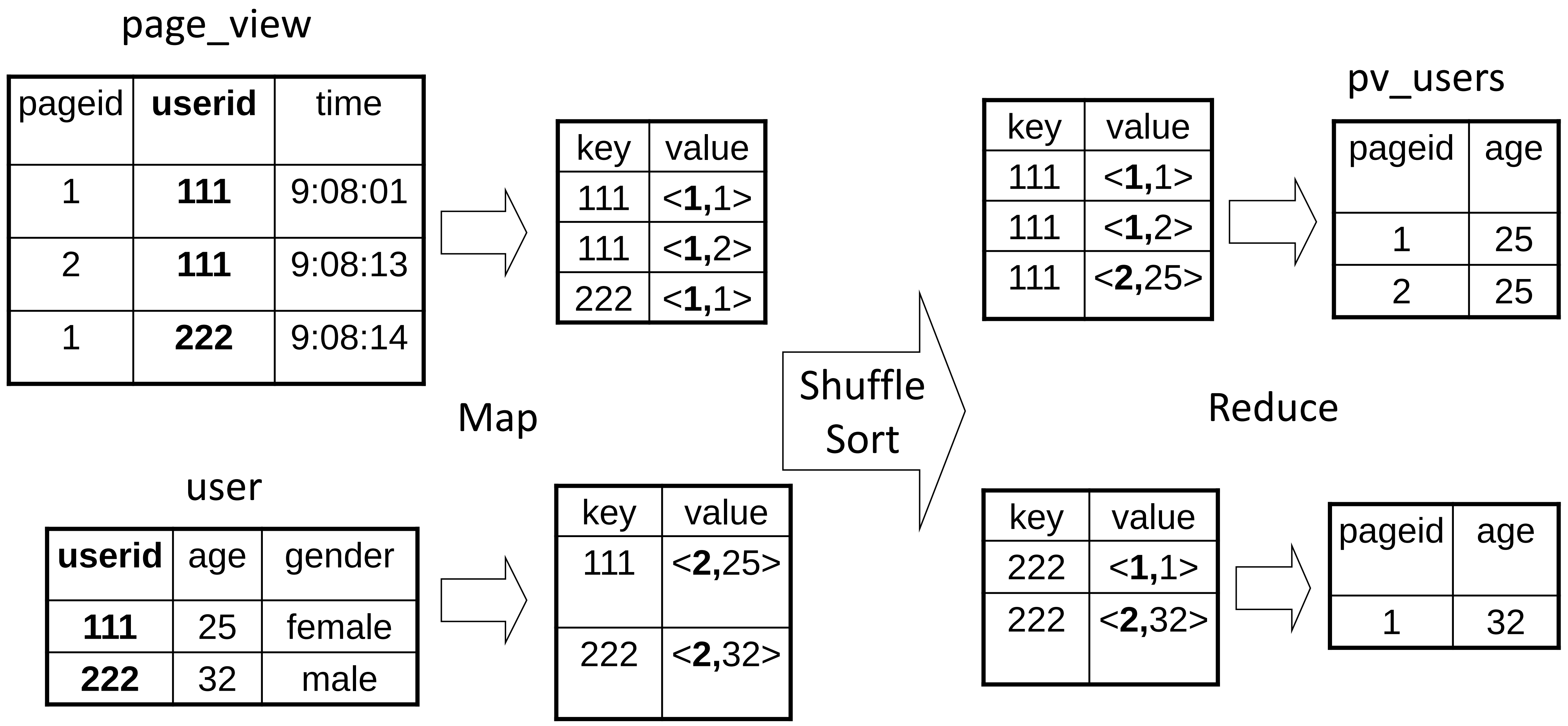
pv_users

pageid	age
1	25
2	25
1	32

HQL:

```
INSERT INTO TABLE pv_users
SELECT pv.pageid, u.age
FROM page_view pv JOIN user u ON (pv.userid = u.userid);
```

Hive QL – Join in MapReduce



page_view

pageid	**userid**	time
1	**111**	9:08:01
2	**111**	9:08:13
1	**222**	9:08:14

user

userid	age	gender
111	25	female
222	32	male

key	value
111	<**1**,1>
111	<**1**,2>
222	<**1**,1>

key	value
111	<**2**,25>
222	<**2**,32>

key	value
111	<**1**,1>
111	<**1**,2>
111	<**2**,25>

key	value
222	<**1**,1>
222	<**2**,32>

pv_users

pageid	age
1	25
2	25

pageid	age
1	32

Hive QL – Group By

pv_users

pageid	age
1	25
2	25
1	32
2	25

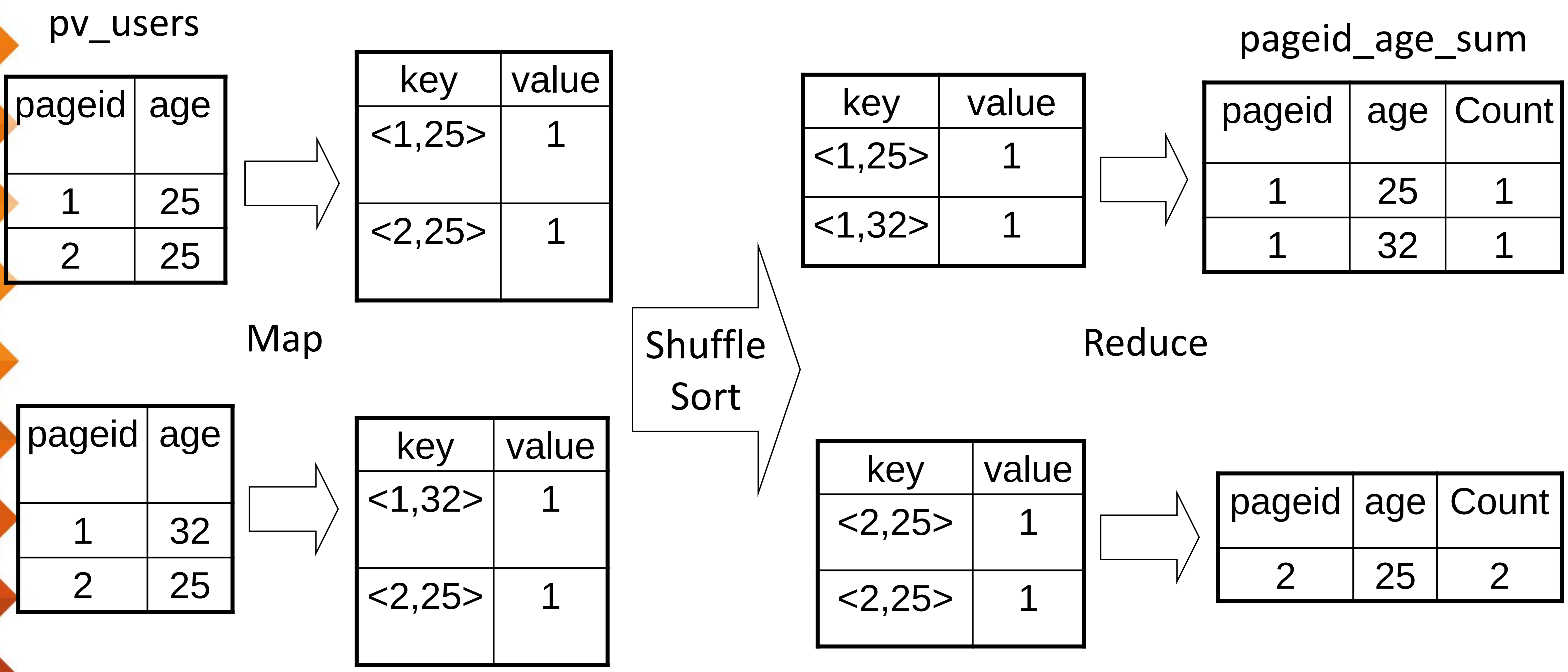
pageid_age_sum

pageid	age	Count
1	25	1
2	25	2
1	32	1

HQL:

```
INSERT INTO TABLE pageid_age_sum
SELECT pageid, age, count(1)
FROM pv_users
GROUP BY pageid, age;
```

Hive QL – Group By in MapReduce



pv_users

pageid	age
1	25
2	25

pageid	age
1	32
2	25

Map

key	value
<1,25>	1
<2,25>	1

key	value
<1,32>	1
<2,25>	1

Shuffle Sort

key	value
<1,25>	1
<1,32>	1

key	value
<2,25>	1
<2,25>	1

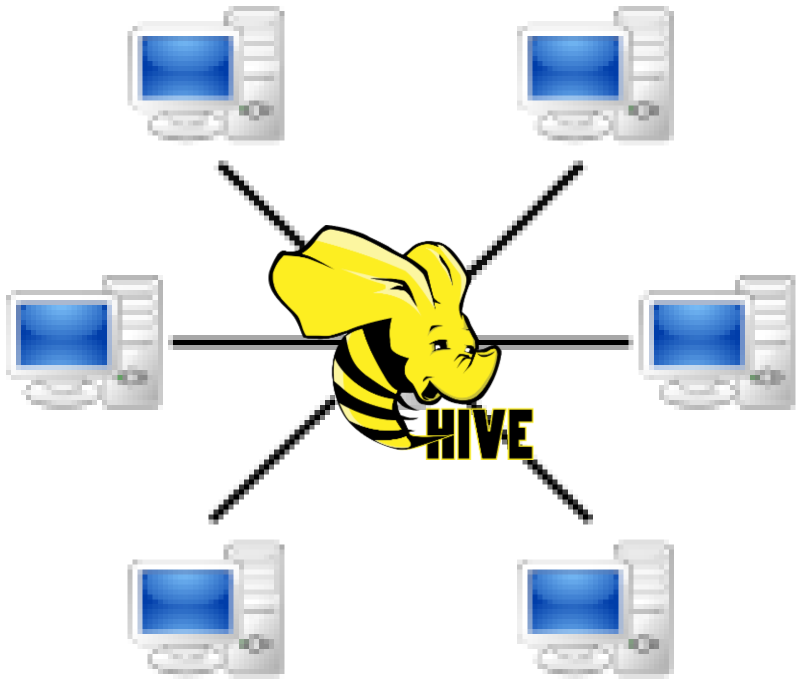
Reduce

pageid_age_sum

pageid	age	Count
1	25	1
1	32	1

pageid	age	Count
2	25	2

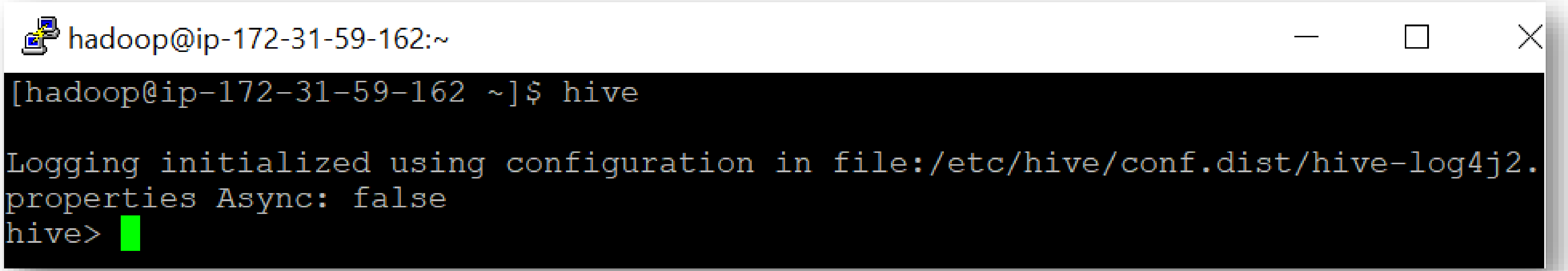
Hive Client



- Hive CLI (Command-Line Interface)
- Web Interface – Hue
- Hive – JDBC (FYI)

Hive CLI (Command-Line Interface)

- Serve as a command line tool for Hive Server
- First client and become legacy

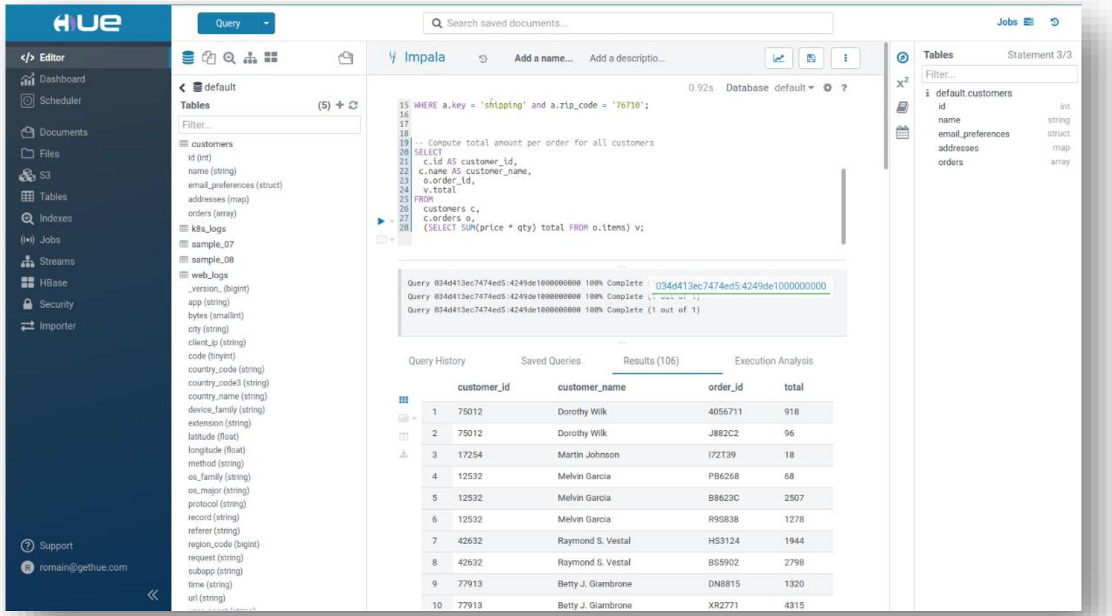


```
[hadoop@ip-172-31-59-162 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.
properties Async: false
hive>
```

Hue

- An open-source SQL Assistant for Databases & Data Warehouses
- Hue (Hadoop User Experience) provides a web front-end
  - to upload and browse data
  - To query tables in Impala and Hive
  - To search and much more
- Makes Hadoop easier to use



```
WHERE a.key = 'shipping' and a.zip_code = '76710';



-- Compute total amount per order for all customers
SELECT
  c.id AS customer_id,
 c.name AS customer_name,
  o.order_id,
  v.total
FROM
  customers c,
  c.orders o,
  (SELECT SUM(price * qty) total FROM o.items) v;
```

```
Query 034d413ec7474ed5:4249de1000000000 100% Complete
Query 034d413ec7474ed5:4249de1000000000 100% Complete
Query 034d413ec7474ed5:4249de1000000000 100% Complete (1 out of 1)
```

	customer_id	customer_name	order_id	total
1	75012	Dorothy Wilk	4056711	918
2	75012	Dorothy Wilk	J882C2	96
3	17254	Martin Johnson	I72T39	18
4	12532	Melvin Garcia	PB6268	68
5	12532	Melvin Garcia	B8623C	2507
6	12532	Melvin Garcia	R9S838	1278
7	42632	Raymond S. Vestal	HS3124	1944
8	42632	Raymond S. Vestal	BS5902	2798
9	77913	Betty J. Giambrone	DN8815	1320
10	77913	Betty J. Giambrone	XR2771	4315

Reference

- Tom White, *A Tour of Apache Hadoop*, Lexeme Ltd.
- Joydeep Sen Sarma and Ashish Thusoo, Facebook Data Team, *Data Warehousing & Analytics on Hadoop – Hive*
- Matei Zaharia, UC Berkeley RAD Lab, *Introduction to MapReduce and Hadoop*
- Getting Started – Apache Hive, Confluence site
- Thusoo et al., *Hive – A Petabyte Scale Data Warehouse Using Hadoop*