Lab 4 Tutorial: Pig Latin

CS4480/CS5488

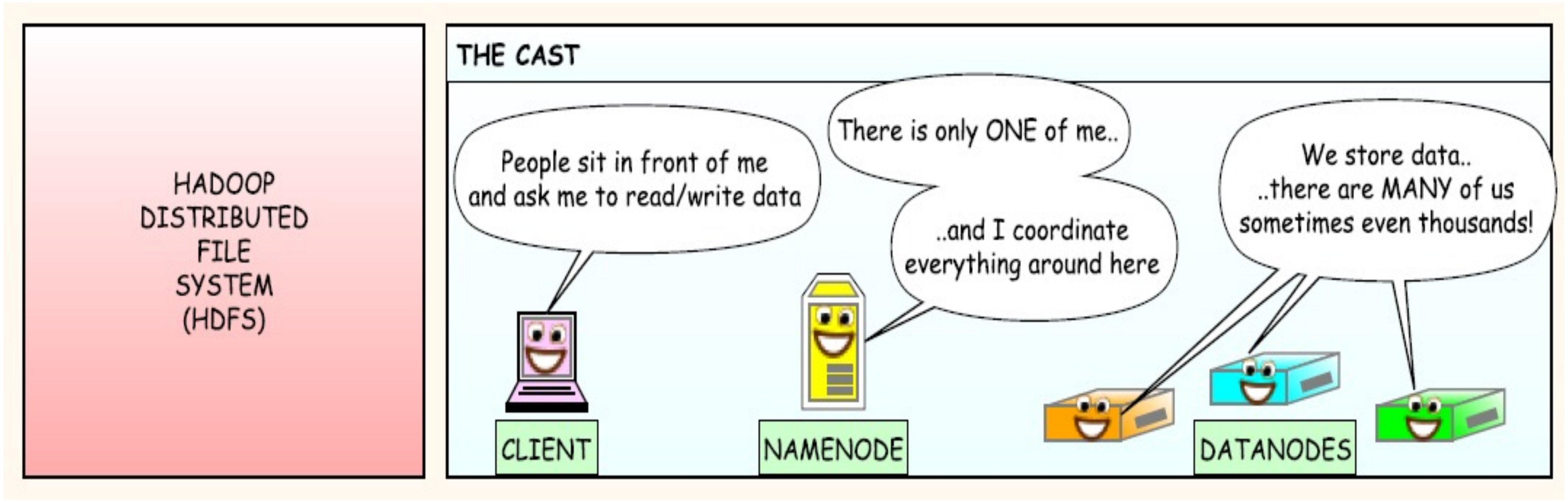
Motivation

- You have data in Hadoop architecture
- How you parallel-handle the data?

For Hadoop, data transformation and aggregation are done in a parallel manner, i.e., all CPUs of the PCs in the cluster work on the instruction for faster processing.

Review Hadoop

- Hadoop is an open-source software platform for **distributed storage** and **distributed processing** of **very large datasets** on **computer clusters** built from **commodity hardware.**


HADOOP DISTRIBUTED FILE SYSTEM (HDFS)
THE CAST
People sit in front of me and ask me to read/write data
There is only ONE of me..
..and I coordinate everything around here
We store data.. ..there are MANY of us sometimes even thousands!
CLIENT
NAMENODE
DATANODES

Hadoop ecosystem

- Hadoop Distributed File System (HDFS)
- MapReduce
- Yet Another Resource Negotiator (YARN)
- Pig
- Hive
- Impala
- HBase

Why Pig?

- Traditional approach:

  Data@SQL Database ← SQL ← Data Process Command

  Programing language

- Hadoop:

  Data@Hadoop Database ← **???** ← Data Process Command

Basically, there are two ways to operate HDFS files:

One is the **command line,** Hadoop provides a set of command-line tools which is similar to Linux file commands;
The other is **Java API**, which uses Hadoop’ s Java library to operate HDFS programmatically document.

??? = Apache Pig

Apache Pig =
  Pig (Hadoop-based Database)
  +
  Pig-Latin (SQL-like Command)

Apache Pig is a tool/platform for analyzing larger data sets and representing them as data streams.

Pig is usually used with Hadoop; we can use Apache Pig to perform all data processing operations in Hadoop.

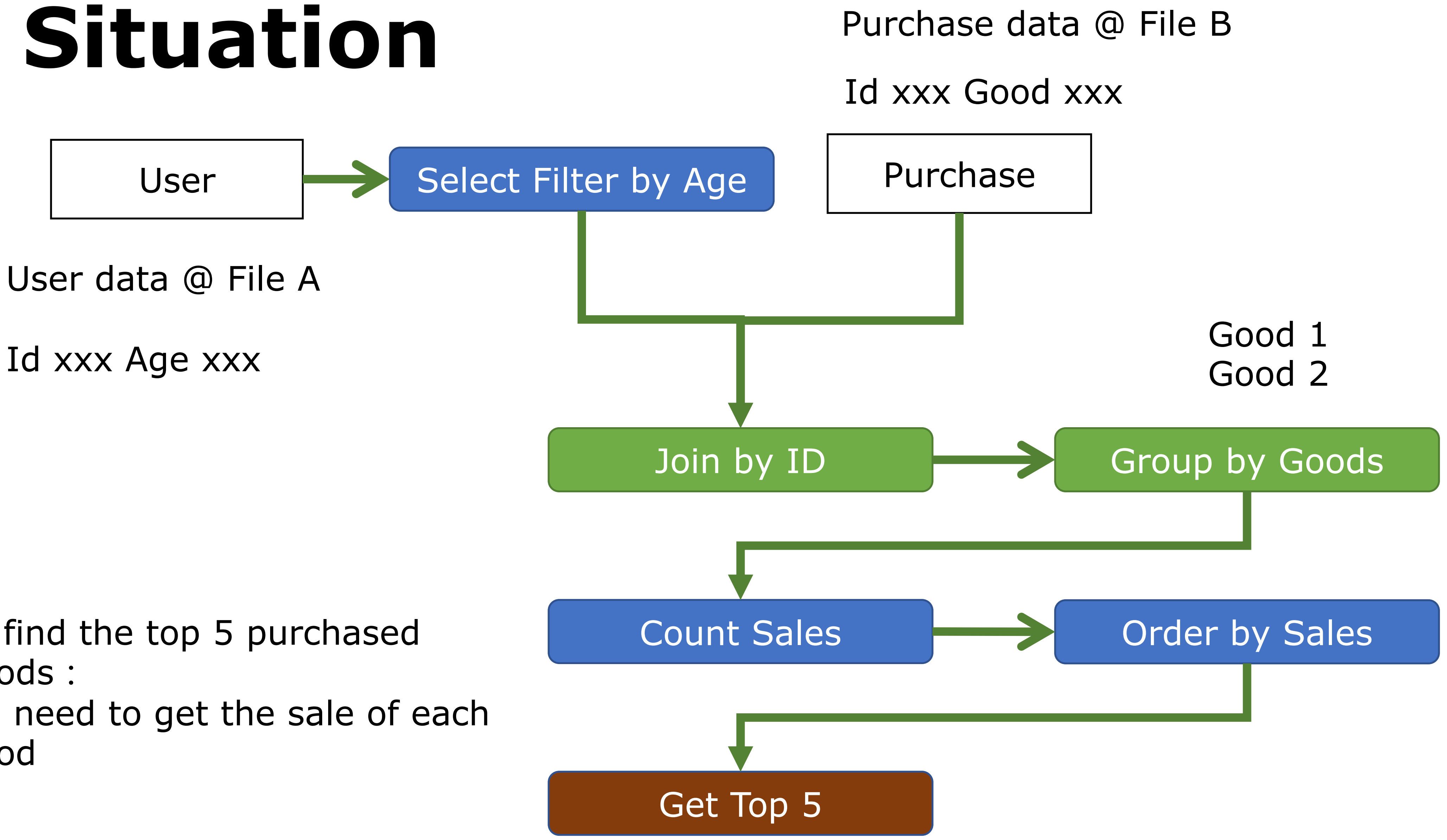
Why Not Use Hadoop ONLY?

E.g. Supermarket Database

- User data @ File A
- Purchase data @ File B

➢ Find the top 5 purchased goods (File B) from users in the range (18, 25) (File A)

Situation



To find the top 5 purchased goods :
we need to get the sale of each good

Situation

MapReduce:

[illegible]

X Hundreds lines of code

Review MapReduce

- MapReduce is a programming model that allows data processing across the entire cluster.
- Mappers: creates the input data, usually a file or directory. The input data is processed to create as few or as many outputs as needed, which is afterward passed to the reducers.
- Reducers: process data from the mapper into something usable. Output from the reducer is saved in the HDFS.
- It’ s a phase known as “shuffle and sort phase” .

??? = Pig Latin

Reduce massive MapReduce codes/commands into several Pig Latin codes/commands

Pig Latin Example

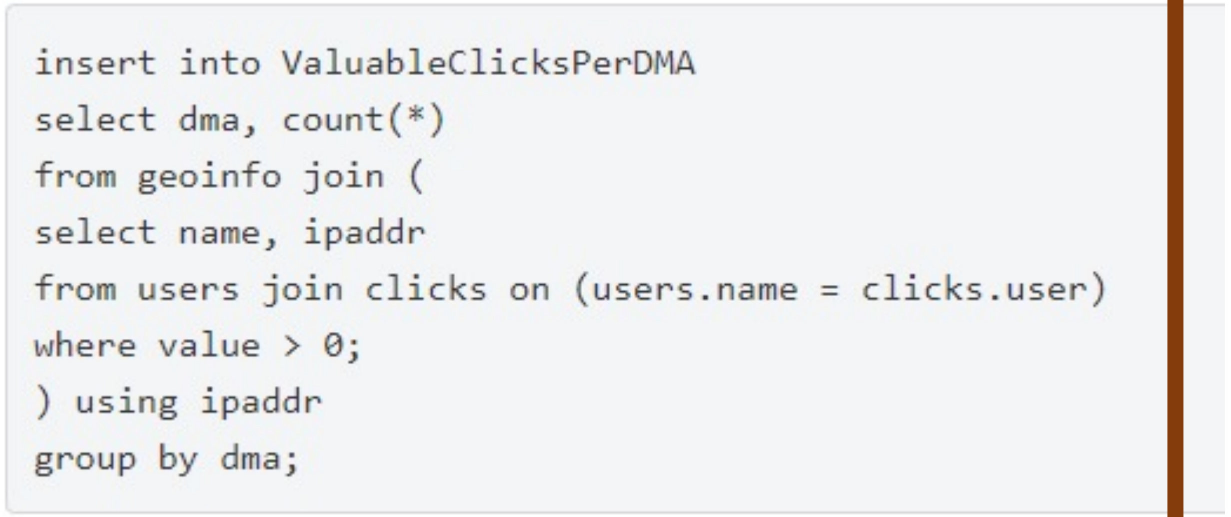
In Pig Latin

```
User = load  'users'  as (ID, age);
Fltrd = filter users by age >= 18 and age <= 25;
Purchases = load  "purchaes"  as (ID, goods);
Jnd = join Fltrd by ID, Purchases by User;
Grpd = group Jnd by goods;
Smmd = foreach Grpd generate group, COUNT(Jnd) as sales;
Srtd = order Smmd by sales desc;
Top5 = limit Srtd 5;
store Top5 into  'top5sales'
```

✓ **Few lines of code**

Pig Latin

- **SQL (Declarative) vs Pig Latin (Procedural)**
- SQL :

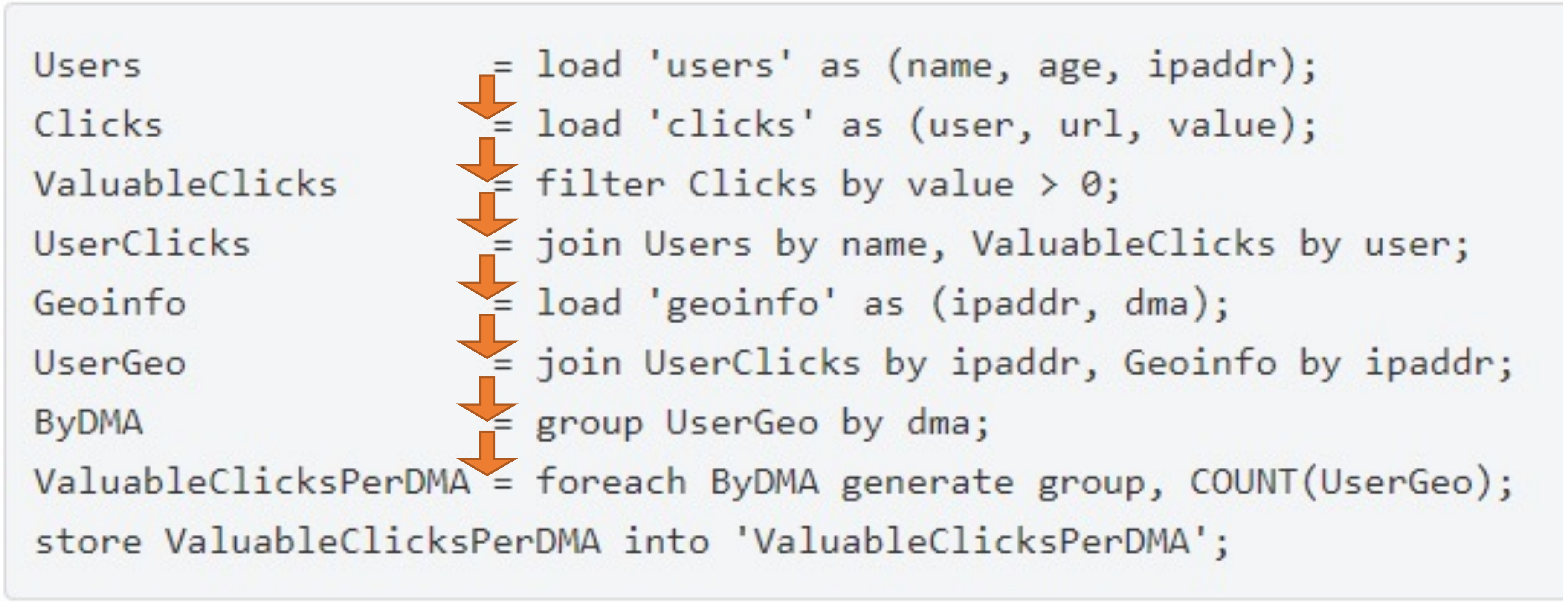


```
insert into ValuableClicksPerDMA
select dma, count(*)
from geoinfo join (
select name, ipaddr
from users join clicks on (users.name = clicks.user)
where value > 0;
) using ipaddr
group by dma;
```

All actions in one big statement

Pig Latin

- **Procedural** for Dataflow as a **Pipeline**



```
Users                = load 'users' as (name, age, ipaddr);
Clicks               = load 'clicks' as (user, url, value);
ValuableClicks       = filter Clicks by value > 0;
UserClicks           = join Users by name, ValuableClicks by user;
Geoinfo              = load 'geoinfo' as (ipaddr, dma);
UserGeo              = join UserClicks by ipaddr, Geoinfo by ipaddr;
ByDMA                = group UserGeo by dma;
ValuableClicksPerDMA = foreach ByDMA generate group, COUNT(UserGeo);
store ValuableClicksPerDMA into 'ValuableClicksPerDMA';
```

Multiple statements

Why Pig for parallel DB?

- Traditional RDBMS (Relational DataBase Management Systems, e.g., SQL) X
  - ➢ Difficult to scale for parallel
  - ➢ Restricted to the relational model

- Hadoop X
  - ➢ One-input (Key-value) two-stage (MapReduce) dataflow
  - ➢ Difficult to optimize
  - ➢ Lack of ad-hoc query supports

Why Pig for parallel DB?

- Pig **O**
  - Address the drawbacks of Parallel RDBMS and MapReduce at the same time

(Pig = Hadoop-based Platform with Pig-Latin)

Pig Latin

- ✓ Like SQL, Pig Latin supports for
  - ➢ a similar set of data types
  - ➢ schema (but optional)
  - ➢ user-defined functions
  - ➢ filesystem operations

Pig Latin

- ✓ Unlike SQL, Pig Latin supports for
  - Sequence of steps (Procedural)
    - ➢ Each step corresponds to an operation
  - High-level transformations
    - ➢ easier to optimize the programming flow
    - ➢ variables available to define the relationships between statements

Why Pig for parallel DB?

✓At Yahoo, 70% MapReduce jobs are written in Pig (https://yahoohadoop.tumblr.com/post/98294444546/comparing-pig-latin-and-sql-for-constructing-data)

Hive

Pig VS Hive

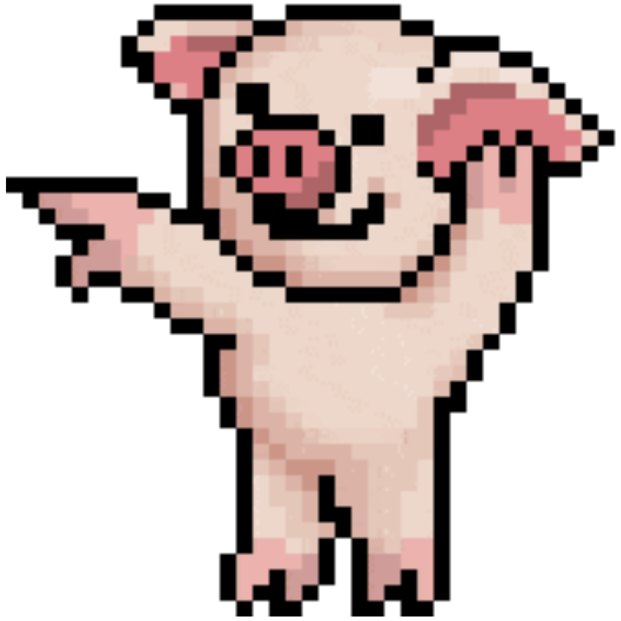
Hive is more suitable for data warehouse tasks. Hive is mainly used for static structures and tasks that require frequent analysis. The similarity between Hive and SQL makes it an ideal intersection of Hadoop and other BI tools.

If you have data warehouse needs and you are good at writing SQL and don't want to write MapReduce jobs, you can use Hive instead.

Pig gives developers more flexibility in the field of large data sets, and allows the development of concise scripts to transform data streams for embedding in larger applications.

Pig is relatively lightweight compared to Hive, and its main advantage is that it can significantly reduce the amount of code compared to directly using Hadoop Java APIs. Because of this, Pig is still attracting a large number of software developers.

Start your Pig Now



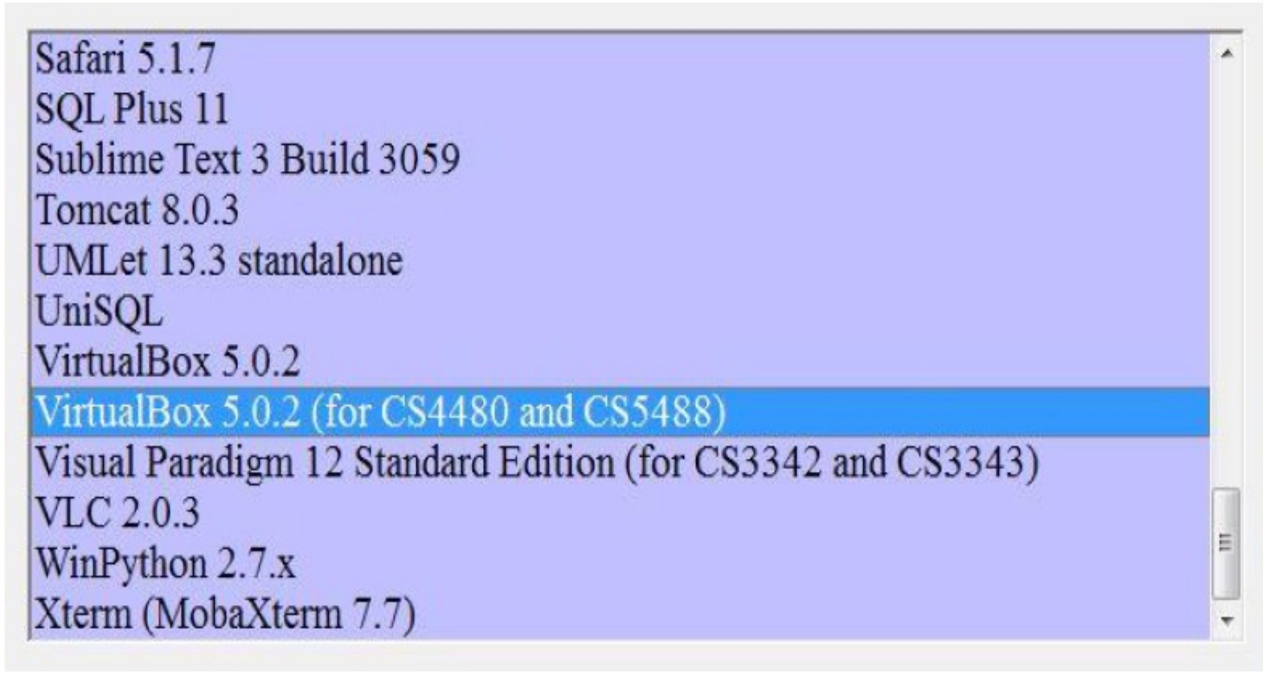
Step 1: Setup

Option 1: Using CS Laptop

Option 2: BYO Laptop

Option 1: Using CS Laptop

- ✓ Open VirtualBox
  - ➢ Click the “VirtualBox 5.0.2 (for CS4480 and CS5488)” item in CSLab Win 10 Apps Menu
- ✓ Run the “Hadoop” VM



Safari 5.1.7
SQL Plus 11
Sublime Text 3 Build 3059
Tomcat 8.0.3
UMLet 13.3 standalone
UniSQL
VirtualBox 5.0.2
VirtualBox 5.0.2 (for CS4480 and CS5488)
Visual Paradigm 12 Standard Edition (for CS3342 and CS3343)
VLC 2.0.3
WinPython 2.7.x
Xterm (MobaXterm 7.7)

Option 2: BYO Laptop

1. Download VirtualBox.
2. Download Hadoop.ova.
3. Import Hadoop.ova into Virtual Box.

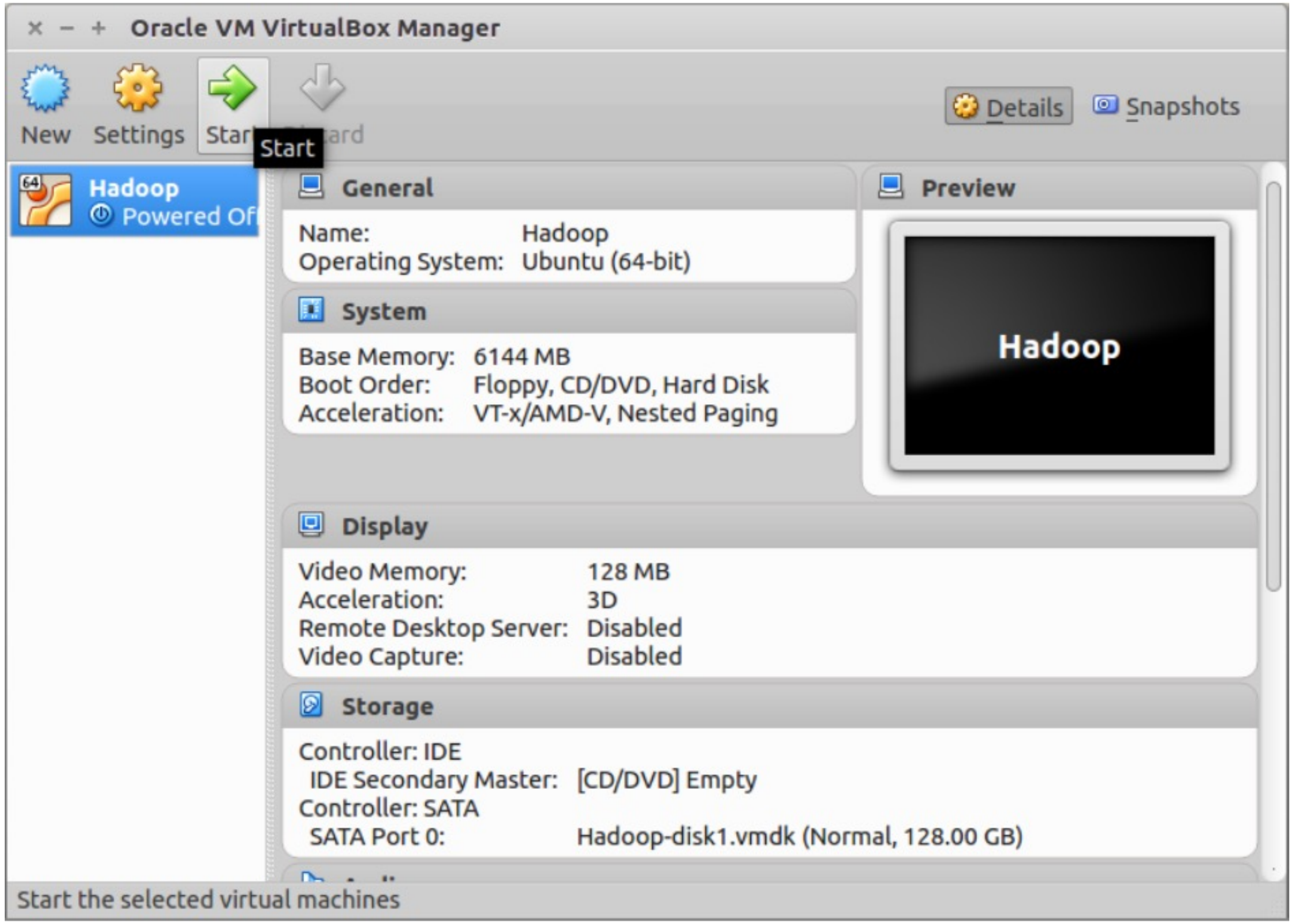
VirtualBox: https://www.virtualbox.org/wiki/Downloads

Hadoop.ova: https://drive.google.com/file/d/1r6Rv5hSyD2a9GCylL6h0ljFyhcDXfEus/view

Setup guidance: in Hadoop tutorial

Step 2: Running VM

✓ Click "Start"
to start the VM



Name: Hadoop
Operating System: Ubuntu (64-bit)

Base Memory: 6144 MB
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

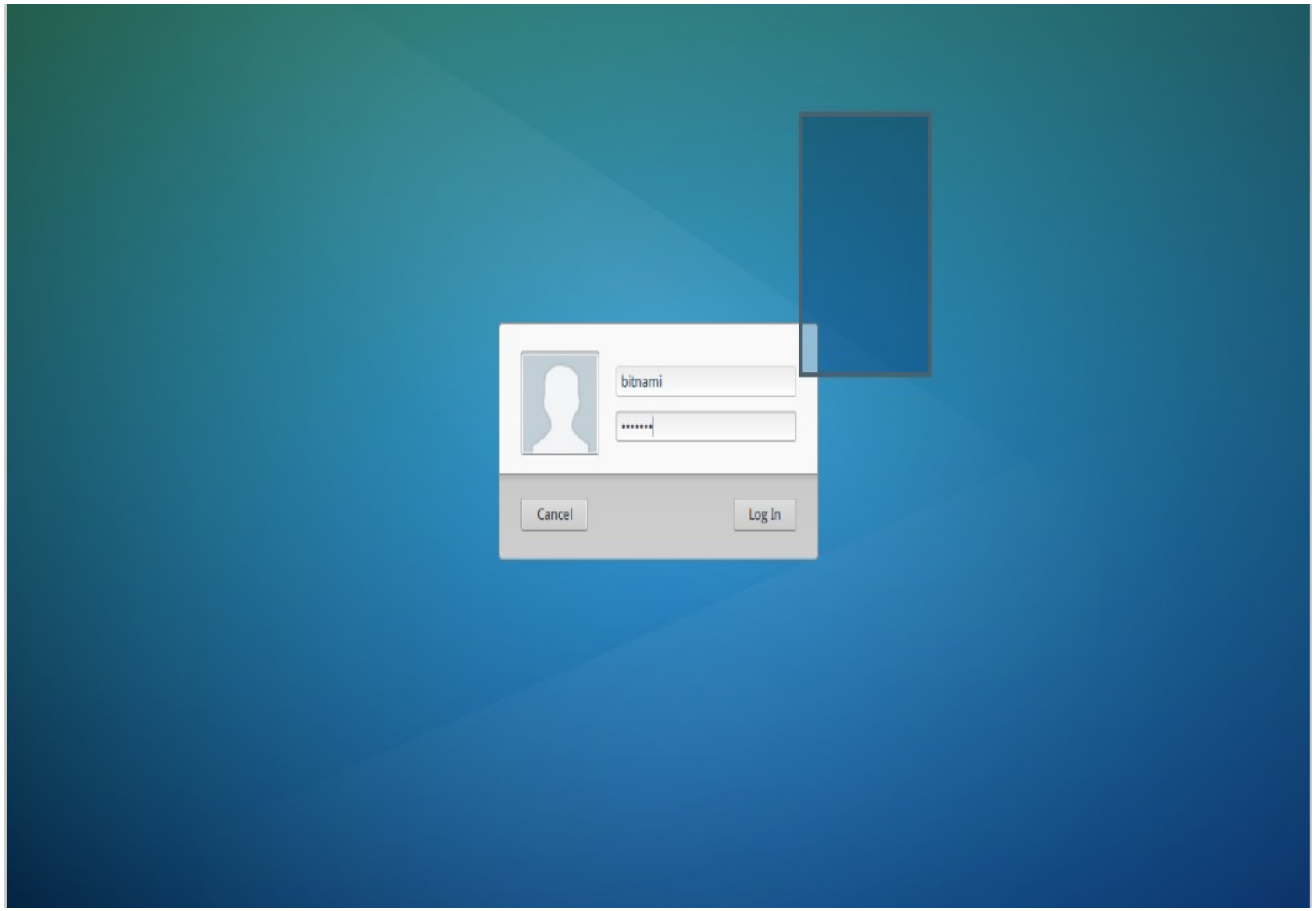
Video Memory: 128 MB
Acceleration: 3D
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: IDE
IDE Secondary Master: [CD/DVD] Empty
Controller: SATA
SATA Port 0: Hadoop-disk1.vmdk (Normal, 128.00 GB)

Start the selected virtual machines

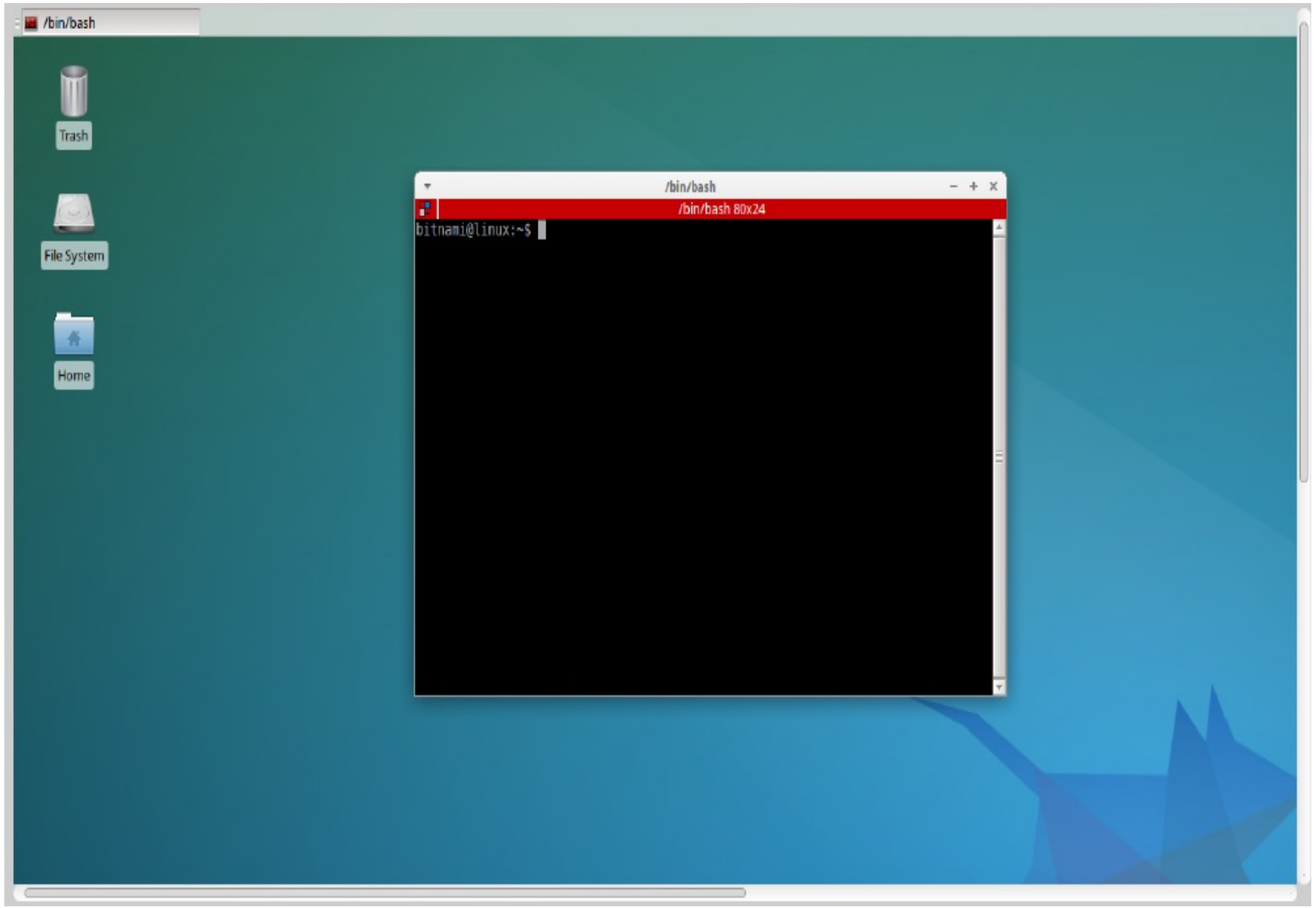
Step 2: Running VM

- ✓ Login with
  - ➢ user: bitnami
  - ➢ password: bitnami



Step 2: Running VM

✓ Open a terminal:
Ctrl+Alt+T

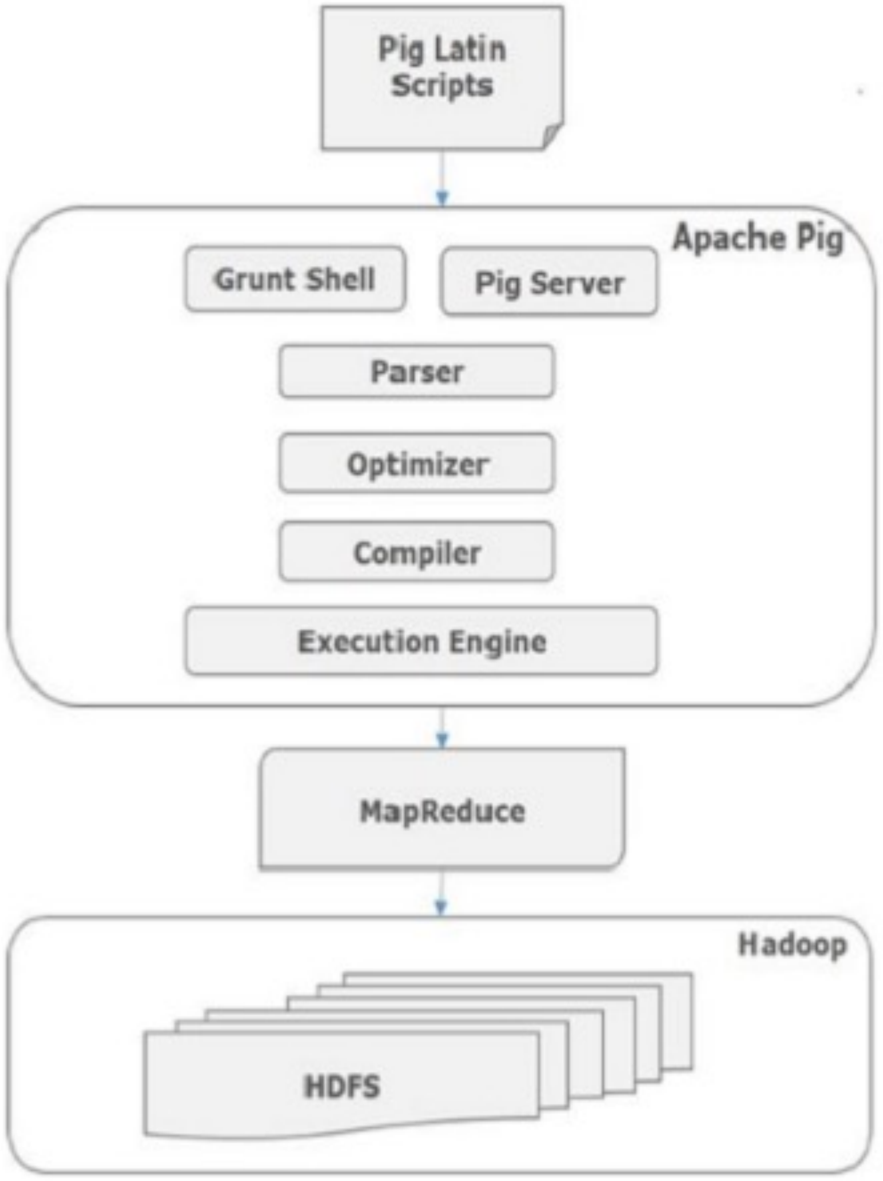


```
bitnami@linux:~$
```

Objective

- Switch on HDFS (Hadoop)
- Learn to Pig Latin

Why need to switch on Hadoop for Pig?



Pig is a large-scale data analysis platform based on Hadoop.

Why need to switch on Hadoop for Pig?

Apache Pig =

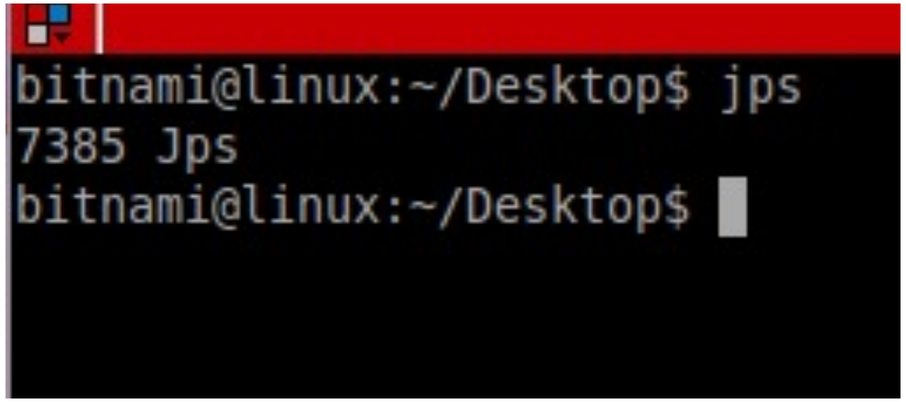
Pig (Hadoop-based Database)

+

Pig-Latin (SQL-like Command)

Step 3: Running Hadoop

✓ Check if all hadoop processes are running by executing jps



```
bitnami@linux:~/Desktop$ jps
7385 Jps
bitnami@linux:~/Desktop$ 
```

What does “jps” mean?

jps

- Java Virtual Machine Process Status Tool
- Check all java application process ID (PID)

We often use the jps command to view java-related processes in the daily production of big data.

Jps is a command that comes with jdk. When you install jdk on your machine and configure jdk into the environment variables of the system, enter jps on the command line to view the current java process.

Step 3: Running Hadoop

✓ If not, execute the following command lines

```
> start-dfs.sh
> start-yarn.sh
> mr-jobhistory-daemon.sh start historyserver
```

You did this step in last lab!

What do these commands mean?

Step 3: Running Hadoop

```
> start-dfs.sh
```

Start the Hadoop DFS daemons, the name node and data node.

```
> start-yarn.sh
```

Start Resource + Node Manager

```
> mr-jobhistory-daemon.sh start historyserver
```

Start the JobHistory Server (Manage Job Log)

Chcek Hadoop status by jps

- When we start the single node cluster, we can see that following processes are up and running:

  Name Node

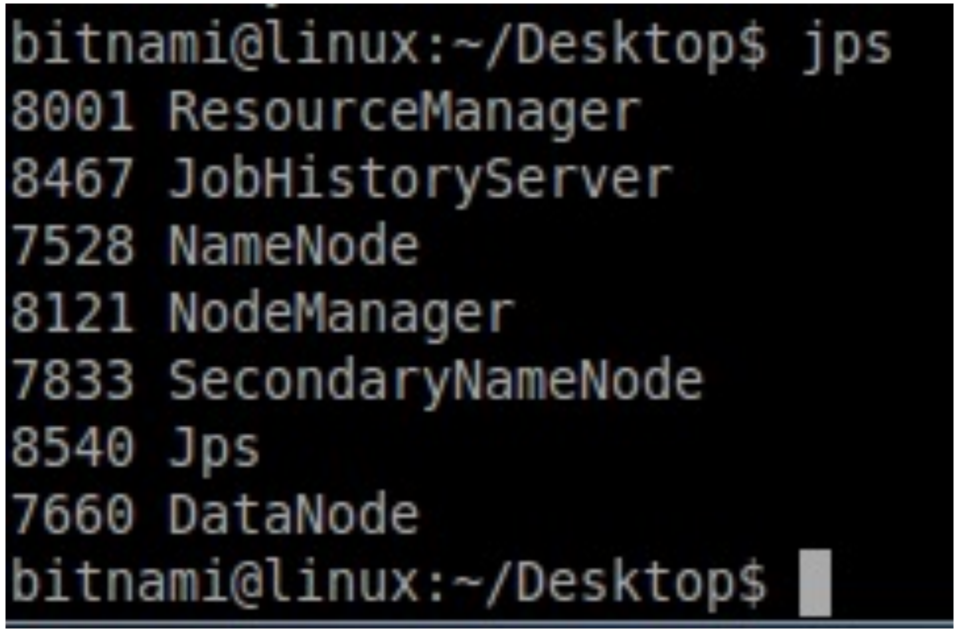
  Data Node

  Resource Manager

  Node Manager

Step 3: Running Hadoop

> jps



```
bitnami@linux:~/Desktop$ jps
8001 ResourceManager
8467 JobHistoryServer
7528 NameNode
8121 NodeManager
7833 SecondaryNameNode
8540 Jps
7660 DataNode
bitnami@linux:~/Desktop$
```

Related to the operation of namenode and datanode under Hadoop

Situation

- Salesmen employee data

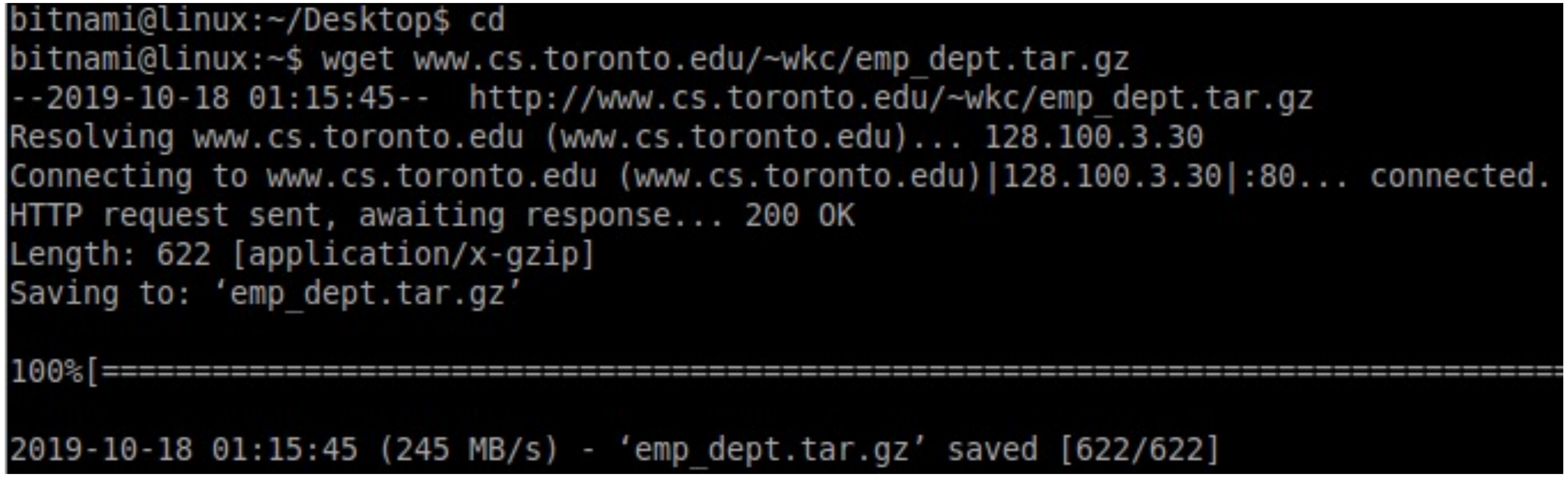
Employee
Department
Sales Grade

Get the Data

- In the terminal, type

> cd
> wget www.cs.toronto.edu/~wkc/emp_dept.tar.gz

to get the data file



```
bitnami@linux:~/Desktop$ cd
bitnami@linux:~$ wget www.cs.toronto.edu/~wkc/emp_dept.tar.gz
--2019-10-18 01:15:45--  http://www.cs.toronto.edu/~wkc/emp_dept.tar.gz
Resolving www.cs.toronto.edu (www.cs.toronto.edu)... 128.100.3.30
Connecting to www.cs.toronto.edu (www.cs.toronto.edu)|128.100.3.30|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 622 [application/x-gzip]
Saving to: ‘emp_dept.tar.gz’

100%[=====================================================================

2019-10-18 01:15:45 (245 MB/s) - ‘emp_dept.tar.gz’ saved [622/622]
```

Decompress the Data File

- Decompress the data file

```
> cd
> tar xzf emp_dept.tar.gz
```

What is tar xzf ?

tar –xvf file.tar.gz // decompress the tar.gz package

Preview the Data File

Change directory to the download path:

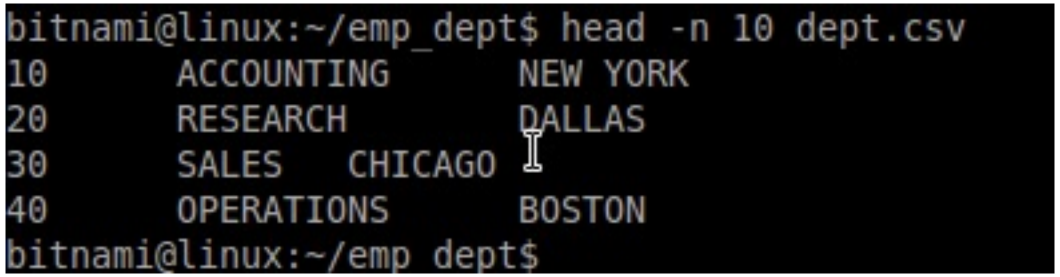
bitnami@linux:~$ cd emp_dept

List all the files in the current directory:

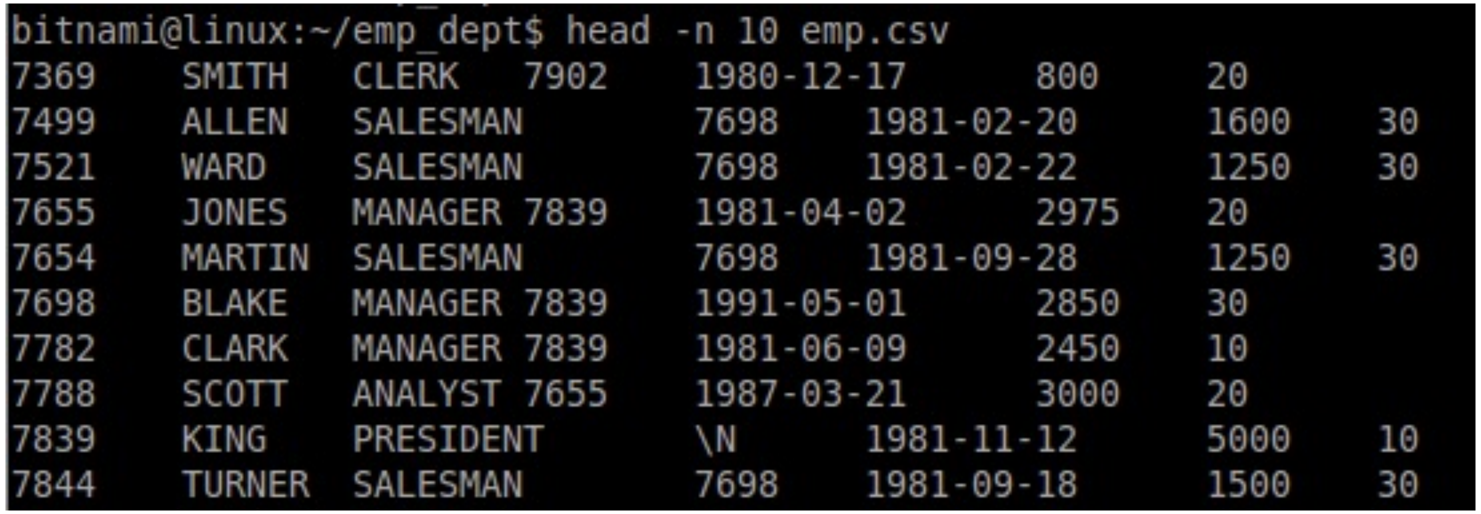
bitnami@linux:~/emp_dept$ ls

See the first 10 lines of the csv file:

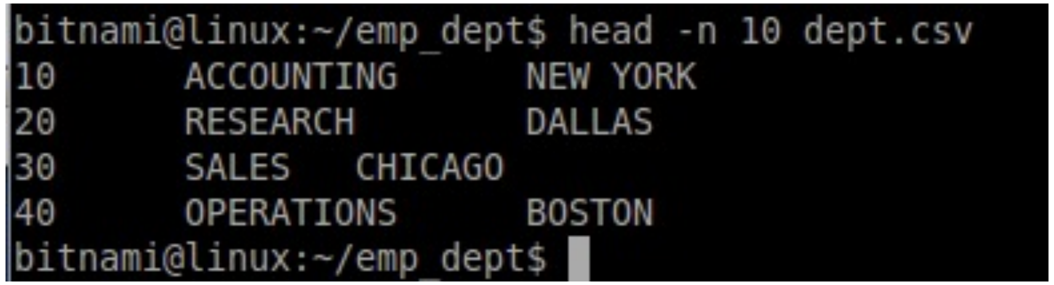
head -n 10 dept.csv



```
bitnami@linux:~/emp_dept$ head -n 10 dept.csv
10      ACCOUNTING      NEW YORK
20      RESEARCH        DALLAS
30      SALES   CHICAGO
40      OPERATIONS      BOSTON
bitnami@linux:~/emp_dept$
```



```
bitnami@linux:~/emp_dept$ head -n 10 emp.csv
7369    SMITH   CLERK   7902    1980-12-17      800     20
7499    ALLEN   SALESMAN        7698    1981-02-20      1600    30
7521    WARD    SALESMAN        7698    1981-02-22      1250    30
7655    JONES   MANAGER 7839    1981-04-02      2975    20
7654    MARTIN  SALESMAN        7698    1981-09-28      1250    30
7698    BLAKE   MANAGER 7839    1991-05-01      2850    30
7782    CLARK   MANAGER 7839    1981-06-09      2450    10
7788    SCOTT   ANALYST 7655    1987-03-21      3000    20
7839    KING    PRESIDENT       \N      1981-11-12      5000    10
7844    TURNER  SALESMAN        7698    1981-09-18      1500    30
```

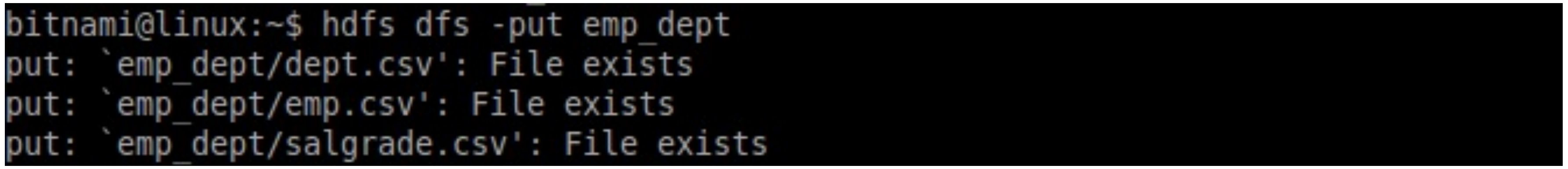


```
bitnami@linux:~/emp_dept$ head -n 10 dept.csv
10      ACCOUNTING      NEW YORK
20      RESEARCH        DALLAS
30      SALES   CHICAGO
40      OPERATIONS      BOSTON
bitnami@linux:~/emp_dept$
```

Transfer the Data to HDFS

- Put the data into the HDFS for Hadoop by issuing the command

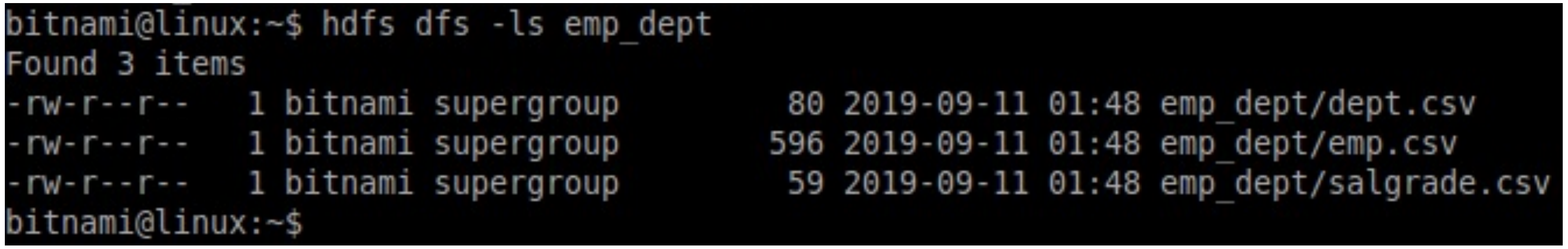
> hdfs dfs -put emp_dept



```
bitnami@linux:~$ hdfs dfs -put emp_dept
put: `emp_dept/dept.csv': File exists
put: `emp_dept/emp.csv': File exists
put: `emp_dept/salgrade.csv': File exists
```

- Then check if the folder exists by using the command

> hdfs dfs -ls emp_dept



```
bitnami@linux:~$ hdfs dfs -ls emp_dept
Found 3 items
-rw-r--r--   1 bitnami supergroup         80 2019-09-11 01:48 emp_dept/dept.csv
-rw-r--r--   1 bitnami supergroup        596 2019-09-11 01:48 emp_dept/emp.csv
-rw-r--r--   1 bitnami supergroup         59 2019-09-11 01:48 emp_dept/salgrade.csv
bitnami@linux:~$
```

Transfer the Data to HDFS

- More info on Hadoop FileSystemShell command

https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html#stat

Step 5: Try Pig

- Go into the Apache Pig environment by typing

```
> pig
```

You should see

```
grunt>
```

grunt? - Node.js (js-based server) package manager

What is the relationship between grunt, pig and javascript?

grunt, pig and javascript

Grunt is a task management tool based on **Node.js**. It can automatically run the tasks you set.

Node.js is an open source, cross-platform runtime environment that can run **JavaScript** on the server side.

Grunt Shell: Run **Pig** code interactively, similar to a python shell.

Step 5: Try Pig

- Load data into the Apache Pig environment:

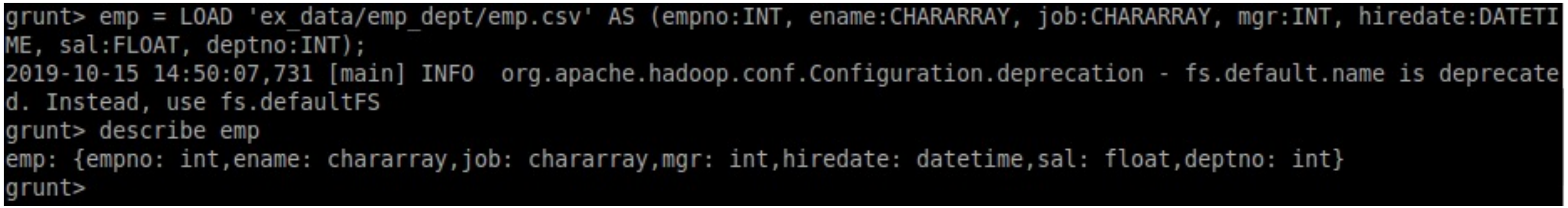
```
> emp = LOAD 'ex_data/emp_dept/emp.csv' AS
(empno:INT, ename:CHARARRAY, job:CHARARRAY, mgr:INT,
hiredate:DATETIME, sal:FLOAT, deptno:INT);
```

What does it mean?

Pig – Schema

emp = LOAD 'ex_data/emp_dept/emp.csv' AS
(empno:INT, ename:CHARARRAY, job:CHARARRAY, mgr:INT, hiredate:DATETIME, sal:FLOAT, deptno:INT);

Try '*describe emp*' to see result



```
grunt> emp = LOAD 'ex_data/emp_dept/emp.csv' AS (empno:INT, ename:CHARARRAY, job:CHARARRAY, mgr:INT, hiredate:DATETI
ME, sal:FLOAT, deptno:INT);
2019-10-15 14:50:07,731 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecate
d. Instead, use fs.defaultFS
grunt> describe emp
emp: {empno: int,ename: chararray,job: chararray,mgr: int,hiredate: datetime,sal: float,deptno: int}
grunt>
```

describe = Describe the schema of a relation (table)

Pig – Schema

Schema in Pig : **Fit Data into the pre-defined tuple {} as data structure**

emp: {empno: int, ename: chararray, job: chararray, …}

Pig – Load

Load operator:

You can use Pig Latin's **LOAD** operator to load data from the **file system (HDFS / Local)** into Apache Pig.

```
Relation_name = LOAD 'Input file path' USING function AS schema;
```

Relation_name - We must mention the relation in which the data is to be stored.
Input file path - We must mention the HDFS directory where the files are stored (In MapReduce mode).
Function- We must choose a function from a set of loading functions provided by Apache Pig.
Schema - We must define the data schema.

Step 5: Try Pig

```
> dept = LOAD 'ex_data/emp_dept/dept.csv' AS
(deptno:INT, dname:CHARARRAY, loc: CHARARRAY);

> salgrade = LOAD 'ex_data/emp_dept/salgrade.csv' AS
(grade:INT, losal:INT, hisal:INT);
```

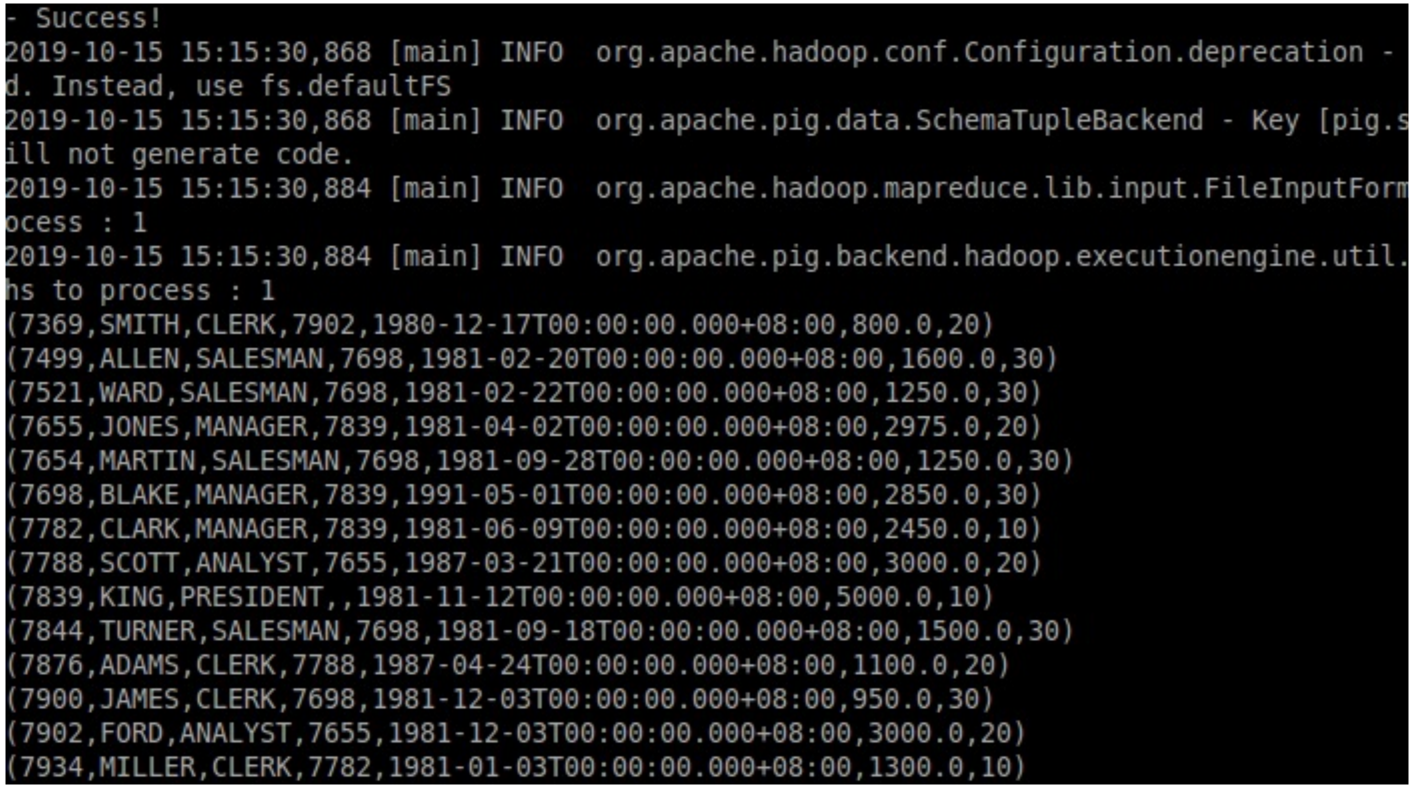
Step 5: Try Pig

- You can type the dump commands for testing

> DUMP emp

It takes a moment to complete. Why?

DUMP C is to output the data in C to the console.

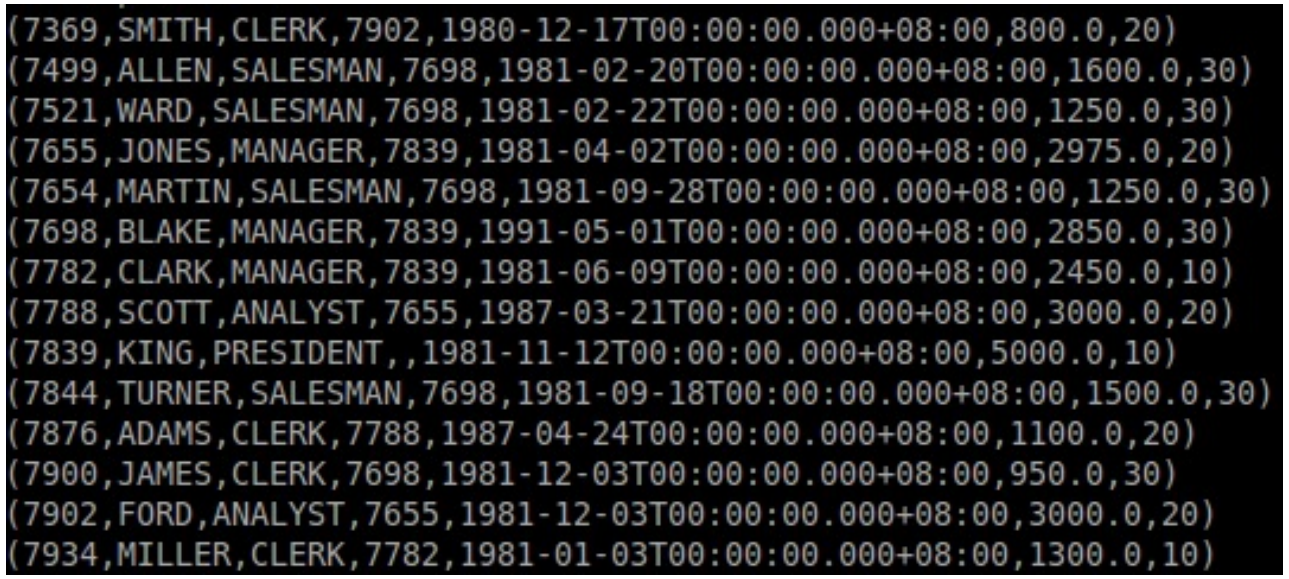


```
- Success!
2019-10-15 15:15:30,868 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
d. Instead, use fs.defaultFS
2019-10-15 15:15:30,868 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.s
ill not generate code.
2019-10-15 15:15:30,884 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputForm
ocess : 1
2019-10-15 15:15:30,884 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.
hs to process : 1
(7369,SMITH,CLERK,7902,1980-12-17T00:00:00.000+08:00,800.0,20)
(7499,ALLEN,SALESMAN,7698,1981-02-20T00:00:00.000+08:00,1600.0,30)
(7521,WARD,SALESMAN,7698,1981-02-22T00:00:00.000+08:00,1250.0,30)
(7655,JONES,MANAGER,7839,1981-04-02T00:00:00.000+08:00,2975.0,20)
(7654,MARTIN,SALESMAN,7698,1981-09-28T00:00:00.000+08:00,1250.0,30)
(7698,BLAKE,MANAGER,7839,1991-05-01T00:00:00.000+08:00,2850.0,30)
(7782,CLARK,MANAGER,7839,1981-06-09T00:00:00.000+08:00,2450.0,10)
(7788,SCOTT,ANALYST,7655,1987-03-21T00:00:00.000+08:00,3000.0,20)
(7839,KING,PRESIDENT,,1981-11-12T00:00:00.000+08:00,5000.0,10)
(7844,TURNER,SALESMAN,7698,1981-09-18T00:00:00.000+08:00,1500.0,30)
(7876,ADAMS,CLERK,7788,1987-04-24T00:00:00.000+08:00,1100.0,20)
(7900,JAMES,CLERK,7698,1981-12-03T00:00:00.000+08:00,950.0,30)
(7902,FORD,ANALYST,7655,1981-12-03T00:00:00.000+08:00,3000.0,20)
(7934,MILLER,CLERK,7782,1981-01-03T00:00:00.000+08:00,1300.0,10)
```

DUMP = Output all records of the relation

Step 5: Try Pig

What the output means?



```
(7369,SMITH,CLERK,7902,1980-12-17T00:00:00.000+08:00,800.0,20)
(7499,ALLEN,SALESMAN,7698,1981-02-20T00:00:00.000+08:00,1600.0,30)
(7521,WARD,SALESMAN,7698,1981-02-22T00:00:00.000+08:00,1250.0,30)
(7655,JONES,MANAGER,7839,1981-04-02T00:00:00.000+08:00,2975.0,20)
(7654,MARTIN,SALESMAN,7698,1981-09-28T00:00:00.000+08:00,1250.0,30)
(7698,BLAKE,MANAGER,7839,1991-05-01T00:00:00.000+08:00,2850.0,30)
(7782,CLARK,MANAGER,7839,1981-06-09T00:00:00.000+08:00,2450.0,10)
(7788,SCOTT,ANALYST,7655,1987-03-21T00:00:00.000+08:00,3000.0,20)
(7839,KING,PRESIDENT,,1981-11-12T00:00:00.000+08:00,5000.0,10)
(7844,TURNER,SALESMAN,7698,1981-09-18T00:00:00.000+08:00,1500.0,30)
(7876,ADAMS,CLERK,7788,1987-04-24T00:00:00.000+08:00,1100.0,20)
(7900,JAMES,CLERK,7698,1981-12-03T00:00:00.000+08:00,950.0,30)
(7902,FORD,ANALYST,7655,1981-12-03T00:00:00.000+08:00,3000.0,20)
(7934,MILLER,CLERK,7782,1981-01-03T00:00:00.000+08:00,1300.0,10)
```

(7369, SMITH, CLERK, 7902, 1980-12-17T00:00:00.000+08:00, 800.0, 20)

(empno:INT, ename:CHARARRAY, job:CHARARRAY, mgr:INT, hiredate:DATETIME, sal:FLOAT, deptno:INT);

Step 6: Using Pig to Retrieve Data

Q15: Get Smith’s employment date:

Step 6: Using Pig to Retrieve Data

Q15: Get Smith’s employment date:

In SQL :

```
SELECT hiredate FROM emp WHERE ename = ‘SMITH’
```

Step 6: Using Pig to Retrieve Data

Q15: Get Smith’s employment date:

In SQL :
SELECT hiredate FROM emp WHERE ename = ‘SMITH’

i. Retrieve emp
ii. Get record with ename = ‘SMITH’
iii. Select hiredate of that record

Step 6: Using Pig to Retrieve Data

In SQL :

i. Retrieve emp
ii. Get record with ename = ‘SMITH’
iii. Select hiredate of that record

In Pig-Latin :

1) result1 = **FILTER** emp **BY** ename == 'SMITH';   (i + ii)
2) result2 = **FOREACH** result1 **GENERATE** hiredate; (iii)
3) DUMP result2;

result1 / result2 is just a defined name of relation (~variable name).

Step 6: Using Pig to Retrieve Data

```
result1 = FILTER emp BY ename == 'SMITH’;
result2 = FOREACH result1 GENERATE hiredate;
DUMP result2;
Output : (1980-12-17T00:00:00.000+08:00)
```

Be careful: Pig is **CASE-SENSITIVE to relation name, data (and some command)**

Case Sensitivity:
The names (aliases) of relations and fields are case-sensitive. The names of Pig Latin functions are case-sensitive. The names of parameters (see Parameter Substitution) and all other Pig Latin keywords (see Reserved Keywords) are cases insensitive.

Pig – FILTER BY & FOREACH GENERATE

Use Filter operator to get the detailed information of the students belonging to New York city:

```
Relation2_name = FILTER Relation1_name BY (condition);

filter_data = FILTER student_details BY city == 'New York';
```

Get the id, age and city values of each student from the relationship student_details, and use the foreach operator to store it in another relationship named foreach_data, as shown below:

```
Relation_name2 = FOREACH Relatin_name1 GENERATE (required data);

foreach_data = FOREACH student_details GENERATE id,age,city;
```

Step 6: Using Pig to Retrieve Data

Q16: Get Ford’s job title

In SQL :

SELECT job FROM emp WHERE ename = ‘FORD’

Step 6: Using Pig to Retrieve Data

Q16: Get Ford’s job title

Pig-Latin :

```
result = FILTER emp BY ename == 'FORD';
result = FOREACH result GENERATE job;
DUMP result;
```

Output : (ANALYST)

Step 7: Guiding for next lab session

- Q17: Get the first employee by the hiredate

SQL :

i. SELECT ename, hiredate FROM emp
ii. ORDER BY hiredate
iii. LIMIT 1

Step 7: Guiding for next lab session

- Q17: Get the first employee by the hiredate

SQL :	Pig-Latin :
i. SELECT ename FROM emp	→ FOREACH … GENERATE …
ii. ORDER BY hiredate	→ **ORDER** … **BY** … (ASC / DESC)
iii. LIMIT 1	→ **LIMIT** … (No. of top result)

```
joinDate1 = FOREACH emp GENERATE ename, hiredate ;
joinDate2 = ORDER joinDate1 BY hiredate ASC ;
earliestJoinDate = LIMIT joinDate2 1 ;
```

```
Output: (SMITH,1980-12-17T00:00:00.000+08:00)
```

Pig – ORDER BY & LIMIT

Sort the relationship in ascending / descending order according to the age of the students, and use the ORDER BY operator to store it in another relationship named order_by_data, as shown below.

```
Relation_name2 = ORDER Relatin_name1 BY (ASC|DESC);

order_by_data = ORDER student_details BY age DESC;
```

The LIMIT operator is used to obtain a limited number of tuples from the relationship.

```
Result = LIMIT Relation_name required number of tuples;

limit_data = LIMIT student_details 4;
```

Step 7: Guiding for next lab session

- Q18: Get the number of employees in each department

SQL :

i. SELECT **<u>Count</u>**(*) from emp

ii. **<u>GROUP BY</u>** deptno

Step 7: Guiding for next lab session

SQL :

i. SELECT **Count(*)** from emp

ii. GROUP BY deptno

Pig-Latin:

(deptno, count(emp))

emp = *
(whole table)

1) GROUP … **BY** …

2) FOREACH … GENERATE **group as** deptno, **COUNT(emp)** **AS** …

Step 7: Guiding for next lab session

SQL :

i. SELECT Count(*) from emp

ii. GROUP BY deptno

iii. INNER JOIN dept **with** deptno (Map deptno to dname)

Pig-Latin:

1) GROUP … BY …

2) FOREACH … GENERATE group as …, COUNT(emp) AS …

3) JOIN … **BY** …, … **BY** …

Step 7: Guiding for next lab session

Pig-Latin:

- empDept = GROUP emp BY deptno ;
- deptEmp = FOREACH empDept GENERATE group AS deptno, COUNT(emp) AS empCnt ;
- jointCnt = JOIN deptEmp BY deptno, dept BY deptno;
- DUMP jointCnt ;

```
Output: (10,3,10,ACCOUNTING,NEW YORK)
        (20,5,20,RESEARCH,DALLAS)
        (30,6,30,SALES,CHICAGO)
```

Pig – GROUP / JOIN BY

The **GROUP** operator is used to group the data in one or more relations. It collects the data having the same key.

```
Group_data = GROUP Relation_name BY age;
```

The **JOIN** operator is used to combine records from two or more relations. While performing a join operation, we declare one (or a group of) tuple(s) from each relation, as keys. When these keys match, the two particular tuples are matched, else the records are dropped. Joins can be of the following types −

```
grunt> result = JOIN relation1 BY columnname, relation2 BY columnname;
```

More info on pig Latin

https://www.tutorialspoint.com/apache_pig/index.htm

Last one:

- More data involved → Longer time to compute
- Write your procedural steps wisely to reduce data scale involved

Thanks

Q&A