MAPREDUCE & PIG TUTORIAL

Hands-on Session

by Suchitra Jayaprakash
suchitra@cmi.ac.in

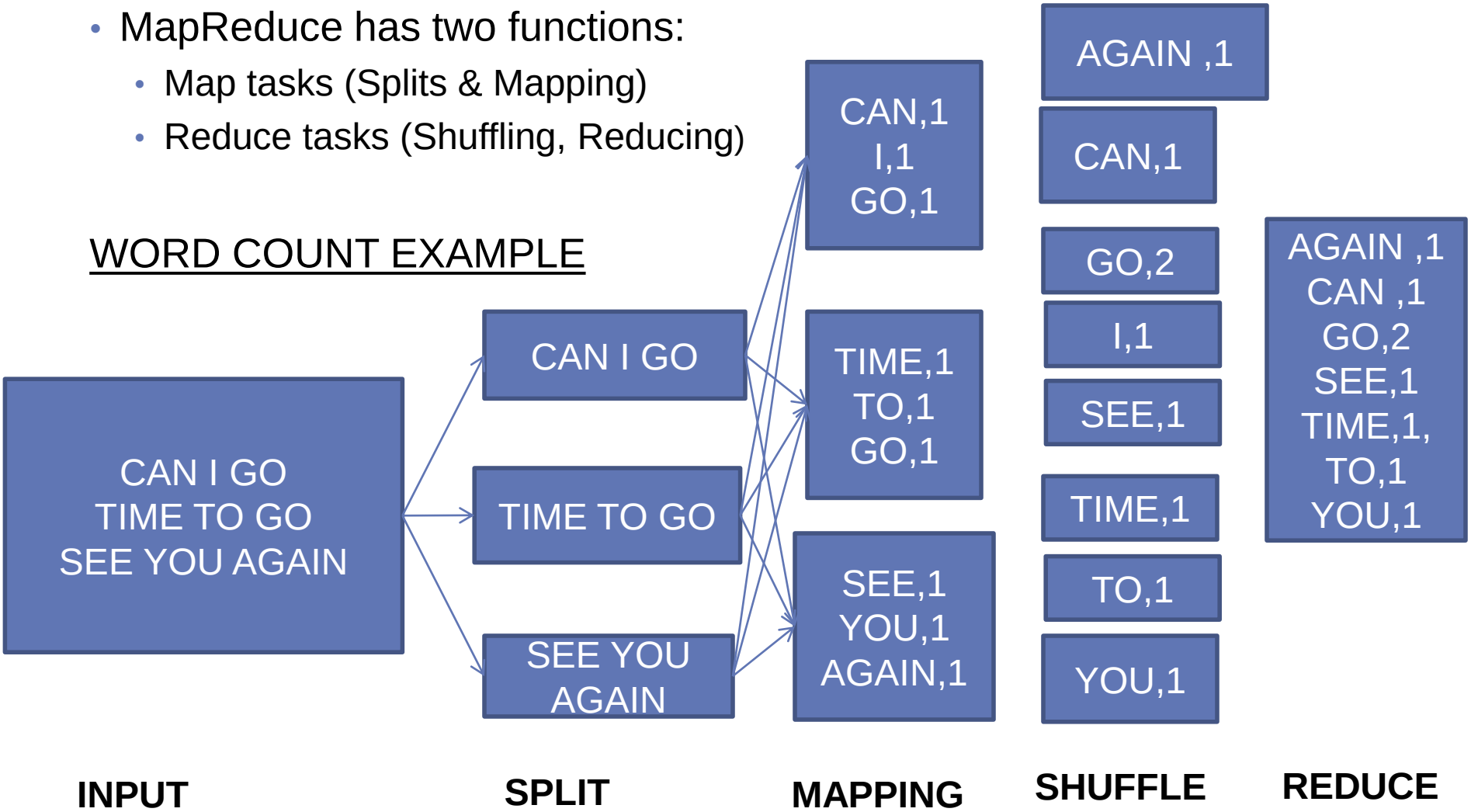
MAPREDUCE

- MapReduce is a programing model. It is specialization of the *split-apply-combine* strategy for data analysis.
- Hadoop's MapReduce implementation is based on Google's paper : MAPREDUCE: SIMPLIFIED DATA PROCESSING ON LARGE CLUSTERS" by Jeffery Dean and Sanjay Ghemawat in 2004.
- Google's proprietary MapReduce system ran on the Google File System (GFS).
- MapReduce leverages HDFS to perform high performance batch processing.

How MapReduce Works?

- MapReduce has two functions:
  - Map tasks (Splits & Mapping)
  - Reduce tasks (Shuffling, Reducing)

WORD COUNT EXAMPLE



MAPREDUCE - Run WordCount

- Start cloudera quick start

docker run --hostname=quickstart.cloudera --privileged=true -t -i --publish-all=true -p 8888:8888 -p 8080:80 -p 50070:50070 -p 8088:8088 -p 50075:50075 -p 8032:8032 -p 8042:8042 -p 19888:19888 cloudera/quickstart /usr/bin/docker-quickstart

Port	Purpose
8088	Yarn (MRv2) - job tracker
8032	ResourceManager
50070	Name node web interface
50075	Data node
8042	NodeManager
19888	JobHistory Server

MAPREDUCE - Run WordCount

- Copy text file to HDFS.

  hadoop fs -mkdir DATA

  docker cp E:/sample1.txt <containerid>:/tmp/sample1.txt
  docker cp E:/sample2.txt <containerid>:/tmp/sample2.txt

  hadoop fs -copyFromLocal /tmp/sample1.txt DATA/sample1.txt
  hadoop fs -copyFromLocal /tmp/sample2.txt DATA/sample2.txt

- Copy WordCount java to Docker .

  docker cp e:/MSc_Datascience/BigDataHadoop/WordCount1.java <containerid>:/tmp/WordCount1.java

- https://docs.cloudera.com/documentation/other/tutorial/CDH5/topics/ht_wordcount1_source.html

MAPREDUCE - Run WordCount

- Compile java class & create jar

mkdir -p build

javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* /tmp/WordCount1.java -d build -Xlint

```
Using CATALINA_PID:     /var/run/solr/solr.pid
Started Impala Catalog Server (catalogd) :                  [  OK  ]
Started Impala Server (impalad):                            [  OK  ]
[root@quickstart /]# hadoop fs -mkdir DATA
[root@quickstart /]# hadoop fs -copyFromLocal /tmp/sample1.txt DATA/sample1.txt
[root@quickstart /]# hadoop fs -copyFromLocal /tmp/sample2.txt DATA/sample2.txt
[root@quickstart /]# mkdir -p build
[root@quickstart /]# javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* /tm
p/WordCount1.java -d build -Xlint
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jaxb-api.jar": no su
ch file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/activation.jar": no
such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jsr173_1.0_api.jar":
 no such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jaxb1-impl.jar": no
such file or directory
4 warnings
[root@quickstart /]#
```

jar -cvf wordcount.jar -C build/ .

```
[root@quickstart /]# jar -cvf wordcount.jar -C build/ .
added manifest
adding: WordCount1$Map.class(in = 2192) (out= 982)(deflated 55%)
adding: WordCount1$Reduce.class(in = 1630) (out= 687)(deflated 57%)
adding: WordCount1.class(in = 1959) (out= 986)(deflated 49%)
[root@quickstart /]#
```

MAPREDUCE - Run WordCount

- Execute Mapreduce Word Count job .

hadoop jar wordcount.jar WordCount1 DATA DATA2

```
[root@quickstart /]# hadoop jar wordcount.jar WordCount1 DATA DATA2
19/12/25 14:37:10 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
19/12/25 14:37:11 INFO input.FileInputFormat: Total input paths to process : 2
19/12/25 14:37:12 INFO mapreduce.JobSubmitter: number of splits:2
19/12/25 14:37:12 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
77284264785_0001
19/12/25 14:37:13 INFO impl.YarnClientImpl: Submitted application application_15
77284264785_0001
19/12/25 14:37:13 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1577284264785_0001/
19/12/25 14:37:13 INFO mapreduce.Job: Running job: job_1577284264785_0001
19/12/25 14:37:26 INFO mapreduce.Job: Job job_1577284264785_0001 running in uber
 mode : false
19/12/25 14:37:26 INFO mapreduce.Job:  map 0% reduce 0%
19/12/25 14:37:43 INFO mapreduce.Job:  map 100% reduce 0%
19/12/25 14:37:52 INFO mapreduce.Job:  map 100% reduce 100%
19/12/25 14:37:53 INFO mapreduce.Job: Job job_1577284264785_0001 completed succe
ssfully
19/12/25 14:37:53 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=96
                FILE: Number of bytes written=340703
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=286
                HDFS: Number of bytes written=39
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
        Job Counters  Number of write operations=2
                Launched reduce tasks=1
                Total time spent by all maps in occupied slots (ms)=27727
                Total time spent by all map tasks (ms)=27727slots (ms)=6612
                Total vcore-seconds taken by all map tasks=27727
                Total megabyte-seconds taken by all map tasks=28392448
        Map-Reduce Frameworkte-seconds taken by all reduce tasks=6770688
                Map output records=8
                Map output materialized bytes=102
```

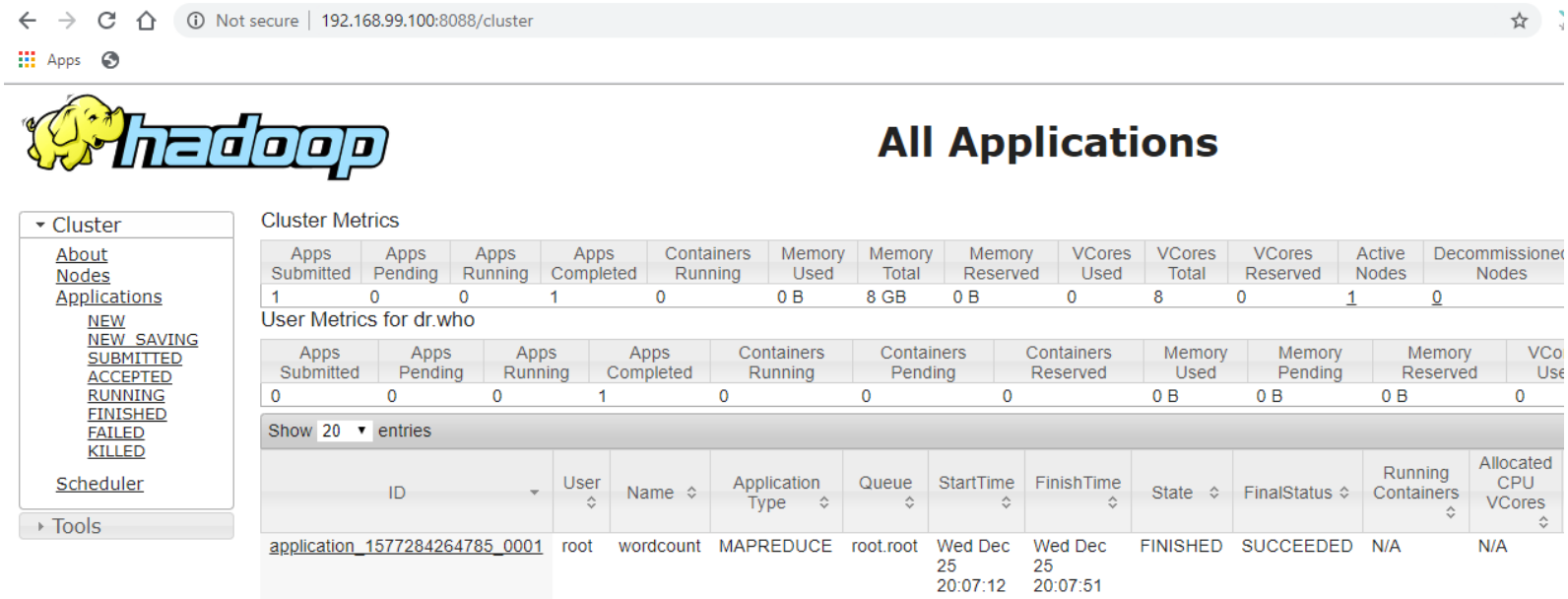
```
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
        Job Counters  Number of write operations=2
                Launched reduce tasks=1
                Total time spent by all maps in occupied slots (ms)=27727
                Total time spent by all map tasks (ms)=27727slots (ms)=6612
                Total vcore-seconds taken by all map tasks=27727
                Total megabyte-seconds taken by all map tasks=28392448
        Map-Reduce Frameworkte-seconds taken by all reduce tasks=6770688
                Map output records=8
                Map output materialized bytes=102
                Input split bytes=246
                Combine input records=0
                Combine output records=0
                Reduce input groups=5
                Reduce shuffle bytes=102
                Reduce input records=8
                Reduce output records=5
                Spilled Records=16
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=278
                CPU time spent (ms)=3250
                Physical memory (bytes) snapshot=720171008
                Virtual memory (bytes) snapshot=4096008192
                Total committed heap usage (bytes)=624427008
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=40
        File Output Format Counters
                Bytes Written=39
[root@quickstart /]#
```

MAPREDUCE - Run WordCount

- Open Yarn web page in browser

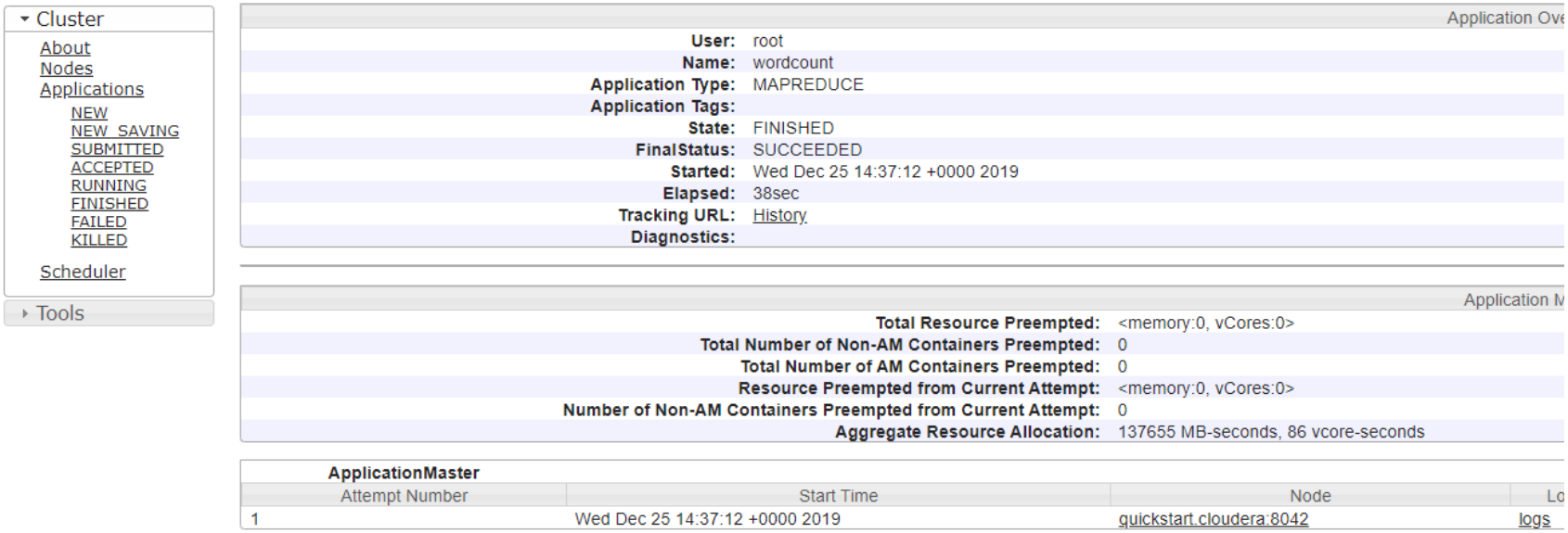
http://192.168.99.100:8088/cluster

http://localhost:8088/cluster



MAPREDUCE - Run WordCount

- Click on JOB ID



- To View Node Manager
- http://192.168.99.100:8042/node/node

MAPREDUCE - Run WordCount

- View Job history

http://192.168.99.100:19888/jobhistory

- Application
- Job
  - Overview
  - Counters
  - Configuration
  - Map tasks
  - Reduce tasks
- Tools

Job Overview

Job Name:	wordcount
User Name:	root
Queue:	root.root
State:	SUCCEEDED
Uberized:	false
Submitted:	Wed Dec 25 14:37:12 UTC 2019
Started:	Wed Dec 25 14:37:25 UTC 2019
Finished:	Wed Dec 25 14:37:51 UTC 2019
Elapsed:	25sec
Diagnostics:	
Average Map Time	13sec
Average Shuffle Time	5sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Wed Dec 25 14:37:18 UTC 2019	quickstart.cloudera:8042	logs

Task Type	Total	Complete
Map	2	2
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	2
Reduces	0	0	1

MAPREDUCE - Run WordCount

- Open DATA2 folder in Namenode to view mapreduce output

http://192.168.99.100:50070/explorer.html#/user/root/DATA2

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/user/root/DATA2 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	0 B	Wed Dec 25 20:07:51 +0530 2019	1	128 MB	_SUCCESS
-rw-r--r--	root	supergroup	39 B	Wed Dec 25 20:07:50 +0530 2019	1	128 MB	part-r-00000

MAPREDUCE - Run WordCount

- Input file

This is sample1 text

This is sample2 text

- Final Output

```
This    2
is  2
sample1 1
sample2 1
text    2
```

Apache PIG

- PIG is an abstraction over MapReduce.
- It uses high level language - Pig Latin.
- PIG Latin is easy to learn if programmer knows procedural language. Programmer can do MapReduce task without having to write complex Java code.
- Pig has many built-in functions & data types for doing operations like joins, filters, ordering, grouping. Thus reducing code length.
- Apache PIG has a internal component called PIG engine.
- PIG engine converts Pig Latin scripts to Map Reduce Task.
- Apache Pig was developed as a research project at Yahoo in 2006. First release of Apache Pig came out in 2008.

Apache PIG – Run Mode

- Pig can be run in various modes:
  - Local mode : Run PIG in local host and file system
  - MapReduce mode : Run PIG on Hadoop cluster and HDFS. It is the default mode.
  - Interactive mode : Run Pig using Grunt shell.
  - Batch mode : Run Pig using PIG script.

	Interactive Mode	Batch Mode
Local Mode	$ pig -x local grunt> <pig Latin statement>	$ pig -x local ScriptFile.pig
MapReduce Mode	$ pig grunt> <pig Latin statement>	$ pig ScriptFile.pig
	$ pig -x mapreduce grunt> <pig Latin statement>	$ pig -x mapreduce ScriptFile.pig

PIG Latin Statement

- Pig Latin statement structure is:

- **LOAD** statement to read data from the file system.
- **Transformation** statements to process the data.
- **DUMP** statement to view results or **STORE** statement to save the results.

- **Load Operator**

  Relation_name = LOAD 'Input file path' USING function as schema;

PIG Latin Statement

Transformation statements :

- FILTER operator to remove unwanted rows of data.
- FOREACH , GENERATE operator to perform data transformations on columns of data.

- GROUP operator to sort data.
- COGROUP, inner JOIN, and outer JOIN operators to group or join data in two or more relations.
- ORDER operator to group data in a single relation.
- UNION operator to merge the contents of two or more relations.
- SPLIT operator to partition the contents of a relation into multiple relations.

Exercise

- Find average sepal length for each flower class in IRIS DATASET

- Reference :

  http://archive.cloudera.com/cdh5/cdh/5/pig-0.12.0-cdh5.15.0/func.html

  http://archive.cloudera.com/cdh5/cdh/5/pig-0.12.0-cdh5.15.0/udf.html

Run PIG

- **Start Cloudera server**

```
docker run --hostname=quickstart.cloudera --privileged=true -t -i  --publish-all=true -p 8888:8888 -p 8080:80 -p 50070:50070 -p 8088:8088 -p 50075:50075 -p 8032:8032 -p 8042:8042 -p 19888:19888 cloudera/quickstart /usr/bin/docker-quickstart
```

- **Copy Text file to docker container**

```
docker cp E:/iris.txt <containerid>:/tmp/iris.txt
```

- **Copy Text file to HDFS**

```
hadoop fs -mkdir DATA
hadoop fs -copyFromLocal /tmp/iris.txt DATA/iris.txt

#view output
hdfs dfs -cat DATA/iris.txt
```

Run PIG

- Start PIG
- type pig and press enter to get PIG command prompt

```
flower = LOAD 'DATA/iris.txt' USING PigStorage(',') as ( sepal_length:int,
sepal_width:int, petal_length:int, petal_width:int, flower_class:chararray);

 DUMP flower;

B = GROUP flower BY flower_class;
DUMP B;

Result = FOREACH B GENERATE flower.flower_class,
AVG(flower.sepal_length);

DUMP Result;
```

OUTPUT

```
Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTIme      AvgMapTime      MedianMa
pTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        A
lias    Feature Outputs
job_1581917396709_0001  1       0       47      47      47      47      n/a     n
/a      n/a     n/a     flower  MAP_ONLY        hdfs://quickstart.cloudera:8020/
tmp/temp1319724132/tmp-899714232,

Input(s):
Successfully read 151 records (4925 bytes) from: "hdfs://quickstart.cloudera:802
0/user/root/DATA/iris.txt"

Output(s):
Successfully stored 151 records (4088 bytes) in: "hdfs://quickstart.cloudera:802
0/tmp/temp1319724132/tmp-899714232"

Counters:
Total records written : 151
Total bytes written : 4088
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1581917396709_0001


2020-02-17 05:50:51,141 [main] WARN  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Encountered Warning ACCESSING_NON_EXISTENT
_FIELD 4 time(s).
2020-02-17 05:50:51,144 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Success!
2020-02-17 05:50:51,161 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-02-17 05:50:51,164 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2020-02-17 05:50:51,169 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Ke
y [pig.schematuple] was not set... will not generate code.
2020-02-17 05:50:51,235 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileI
```

OUTPUT

```
(6,3,5,2,Iris-virginica)
(6,3,5,1,Iris-virginica)
(6,3,4,1,Iris-virginica)
(6,3,5,2,Iris-virginica)
(6,3,5,2,Iris-virginica)
(6,3,5,2,Iris-virginica)
(5,2,5,1,Iris-virginica)
(6,3,5,2,Iris-virginica)
(6,3,5,2,Iris-virginica)
(6,3,5,2,Iris-virginica)
(6,2,5,1,Iris-virginica)
(6,3,5,2,Iris-virginica)
(6,3,5,2,Iris-virginica)
(5,3,5,1,Iris-virginica)
(,,,,)
grunt> B = GROUP flower BY flower_class;
2020-02-17 05:53:12,148 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-02-17 05:53:12,150 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
grunt> Result = FOREACH B GENERATE flower.flower_class, AVG(flower.sepal_length)
;
grunt> DUMP Result;
2020-02-17 05:53:42,813 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: GROUP_BY
2020-02-17 05:53:42,826 [main] INFO  org.apache.pig.newplan.logical.optimizer.Lo
gicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateFor
EachColumnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptim
izer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimiz
er, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter],
RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterOptimizer]}
2020-02-17 05:53:42,909 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? fal
se
2020-02-17 05:53:42,959 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2020-02-17 05:53:42,962 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2020-02-17 05:53:43,080 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Con
necting to ResourceManager at /0.0.0.0:8032
2020-02-17 05:53:43,097 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig script settings are added to the job
2020-02-17 05:53:43,190 [main] INFO  org.apache.pig.backend.hadoop.executionengi
```

OUTPUT

```
2020-02-17 05:56:39,541 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-02-17 05:56:39,544 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2020-02-17 05:56:39,551 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Ke
y [pig.schematuple] was not set... will not generate code.
2020-02-17 05:56:39,583 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileI
nputFormat - Total input paths to process : 1
2020-02-17 05:56:39,585 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.util.MapRedUtil - Total input paths to process : 1
({(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-se
tosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iri
s-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),
(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-seto
sa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-
setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(I
ris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa
),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-se
tosa),(Iris-setosa),(Iris-setosa),(Iris-setosa),(Iris-setosa)},4.6)
({(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virg
inica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris
-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),
(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virgin
ica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-v
irginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(I
ris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginic
a),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-vir
ginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iri
s-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica),(Iris-virginica)
,(Iris-virginica),(Iris-virginica),(Iris-virginica)},6.08)
({(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-
versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicol
or),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iri
s-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versic
olor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(I
ris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-vers
icolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),
(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-ve
rsicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor
),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-
versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicolor),(Iris-versicol
or),(Iris-versicolor)},5.48)
({()},)
grunt>
```

USER DEFINED FUNCTION

- **Find Volume Weighted Average Price of a stock per year.**

Date	Adj Close	Close	High	Low	Open	Volume
31-12-2007	92.64	92.64	94.37	92.45	93.81	5755200
02-01-2008	96.25	96.25	97.43	94.7	95.35	13858700
03-01-2008	95.21	95.21	97.25	94.52	96.06	9122500
04-01-2008	88.79	88.79	93.4	88.5	93.26	10270000

VWAP= sum(Price * Volume)/ sum(Volume)

It is a measure of the average price at which a stock is traded over the trading horizon.

USER DEFINED FUNCTION

- **Find Volume Weighted Average Price of a stock**

Copy content to docker:

docker cp E:/MSc_Datascience/BigDataHadoop/Slides/stock.csv <containerid>:/tmp/stock.csv

docker cp E:/MSc_Datascience/BigDataHadoop/Slides/pig_UDF.py <containerid>:/tmp/pig_UDF.py

docker cp E:/MSc_Datascience/BigDataHadoop/Slides/piggybank-0.15.0.jar <containerid>:/tmp/piggybank-0.15.0.jar

hadoop fs -mkdir DATA

hadoop fs -copyFromLocal /tmp/stock.csv DATA/stock.csv

USER DEFINED FUNCTION

Execute in PIG Interface:

```
REGISTER '/tmp/pig_UDF.py' using jython as myudfs;
REGISTER '/tmp/piggybank-0.15.0.jar';

records = LOAD 'DATA/stock.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX',
'SKIP_INPUT_HEADER') AS
(Date:datetime,AdjClose:int,Close:int,High:int,Low:int,Open:int,Volume:int);

record= foreach records GENERATE Volume, Close, GetYear(Date) as Year;

yearData = GROUP record BY Year;

 vol_wt_avg_price = FOREACH yearData GENERATE group,
myudfs.calVWAP(record.Year,record.Volume,record.Close);

STORE vol_wt_avg_price INTO '/user/root/PRICE';

hadoop fs -cat /user/root/PRICE/part-r-00000
```

QUIZ 4

```
emp_data.txt
001,Mary,32,Delhi
002,Ram,33,Bangalore
```

Q) Employee data file which contains details such as id, name , age and city is stored in HDFS . Data transformation is performed using PIG script below.

```
emp_data = LOAD 'emp_data.txt' USING PigStorage(',')as (id:int, name:chararray, age:int, city:chararray);
substring_data = FOREACH emp_data GENERATE (id,name), SUBSTRING (city, 0, 3);
Dump substring_data ;
```

What is the output of the code?

A) (Del)
(Ban)

B) (1,Del)
(2, Ban)

C) (001,Mary,Del)
(002,Ram, Ban)

D) ((1,Mary),Del)
((2,Ram), Ban)

THANK YOU