PIG Lab Session: Instructions

1. The environment in which Pig Latin commands are executed :- Currently there is support for Local and Hadoop modes
2. Pig compiler converts Pig Latin to MapReduce
3. Think of Pig as a "Pig Latin" compiler, development tool and executor

Modes:

pig – Connects to distributed parallel environment; HDFS

pig -x local (Connects to file system at: file:///)

grunt is an interactive shell for executing Pig Commands, it is started when script file is NOT provided. It can execute scripts from grunt via exec command.

Building blocks :

1. Field - piece of data
2. Tuple - ordered set of fields, represented with "(" and ")" o (10.4, 5, word, 4, field1)
3. Bag - collection of tuples, represented with "{" and "}" o { (10.4, 5, word, 4, field1), (this, 1,blah) }

Analogy of Data structures in PIG to Relational Database :

Bag is a table in the database Tuple is a row in a table

Key Points:

1. Pig is referred to as a Data Flow Languages
2. Most preferred tool for ETL
3. Simple and easy to express
4. Lazy evaluation

Agenda:

In this lab following operators and functions will be demonstrated:

1. **LOAD** : Load data from HDFS/Local into pig environment
2. **PigStorage**: Function to load delimited files
3. **AS** : Operator to provide the schema/ change the existing schema
4. **FILTER**: Apply some filtering criteria on the relations
5. **FOREACH**: Iterate over each row and apply the function called
6. **GROUP**: Gather the data on which columns are grouped into. Number of rows after this operation will be equal to number of distinct values in the selected group.
7. **ORDER**: Sort the relation ascending/descending based on the column selected
8. **COGROUP**: Group operation applied on two relations
9. **JOINS**: Joins to combine two datasets on similar fields
10. **DISTINCT**: select the unique values
11. **SPLIT**: Split the existing relation into multiple relations

12. **UNION**: row binding two datasets. Both the datasets are expected to have same schema in the same order
13. **PARALLEL**: Increase the parallelism for the selected operation
14. **LIMIT**: Limit the number of rows to be selected
15. **STORE**: Invoke the map reduce program and stores the results in the provided folder
16. **DUMP**: Invokes map reduce program and prints the results on screen
17. Sample dataflow
18. **DESCRIBE**: describe of schema of selected relation
19. **ILLUSTRATE**: present the logical data flow
20. **EXPLAIN**: Present the logical and physical map reduce plan

Pre-Lab Activties:

1. Create a directory in your local home folder on the name of pigpractice
   mkdir pigpractice
2. Copy the two text files into pigpractice directory using SCP
3. Copy these files to HDFS:
   hdfs dfs -mkdir pig
   hdfs dfs -put practicefile.txt pig
   hdfs dfs -ls pig
4. Enter into pig shell
   pig

PIG Lab:

5. In pig, HDFS commands can be executed:
   ls
   ls pig
   cat pig/practicefile.txt
6. **LOAD**
   retail = LOAD 'pig/practicefile.txt' USING PigStorage(',');
   describe retail;
   retail = LOAD 'pig/practicefile.txt' USING PigStorage(',') AS
   (region:chararray,cust:chararray,sale:int,trcount:int);
   describe retail;
   dump retail;
7. **FILTER**
   region1cust = FILTER retail BY region=='R1';
   STORE region1cust INTO 'pig/region1cust';
   ls pig/region1cust;
   cat pig/region1cust/part-m-00000;
8. **FOREACH**
   -- refer by name
   tranavg = FOREACH retail GENERATE region,cust,(float)trcount/sale as value;
   --refer by position

```
tranavg = FOREACH retail GENERATE $0,$1,(float)$3/$2 as value;
dump tranavg;
```

9. **GROUP**

```
groupregion = GROUP retail BY region;
describe groupregion;
regionsummary = FOREACH groupregion GENERATE group as region, SUM(retail.sale) as
totalsales, COUNT(retail) as recordcount, (float)SUM(retail.trcount)/SUM(retail.sale) as value;
describe regionsummary;
dump regionsummary;
```

10. **ORDER**

```
sortsales = ORDER retail BY sale;
DUMP sortsales;
sortsales = ORDER retail BY sale desc;
DUMP sortsales;
```

11. **LOAD**

```
# in other terminal
hadoop fs -put custinfo pig
# in pig
ls pig
custinfo = LOAD 'pig/custinfo' USING PigStorage(',') AS (cust,name,sale);
describe custinfo;
dump custinfo;
```

12. **COGROUP**

```
cogroup_result = COGROUP retail BY cust,custinfo BY cust;
describe cogroup_result;
dump cogroup_result;
```

13. **JOINS**

```
join_result = JOIN retail BY cust, custinfo BY cust;
descrine join_result;
explain join_result;
dump join_result;
```

14. **DISTINCT**

```
region = FOREACH retail GENERATE region;
region_distinct = DISTINCT region;
DESCRIBE region_distinct;
DUMP region_distinct;
```

15. **SPLIT**

```
SPLIT retail INTO retail_r1 IF region=='R1', retail_r2 IF region=='R2';
```

16. **UNION**

```
custinfo1 = FOREACH custinfo GENERATE cust,name,sale;
unioncust = UNION custinfo,custinfo1;
```

17. **PARALLEL**

```
groupregion = GROUP retail BY region parallel 8;
```

```
regionsummary = FOREACH groupregion GENERATE group as region, SUM(retail.sale) as
totalsales, COUNT(retail) as recordcount, (float)SUM(retail.trcount)/SUM(retail.sale) as value
parallel 8;
DESCRIBE regionsummary;
DUMP regionsummary;
EXPLAIN regionsummary;
DUMP regionsummary;
STORE regionsummary INTO 'pig/regionsummary'
```

18. **LIMIT**

```
top3records = LIMIT retail 2;
dump top3records;
```

19. **Each region, top 2 customers based on sale. After each command use DESCRIBE and DUMP**

```
retail = LOAD 'pig/practicefile' USING PigStorage(',') AS
(region:chararray,cust:chararray,sale:int,trcount:int);
region_cust = GROUP retail by (region,cust);
region_cust_sales = FOREACH region_cust GENERATE FLATTEN(group), SUM(retail.sale);
region_cust_sales = FOREACH region_cust_sales GENERATE $0 as region, $1 as cust, $2 as
totalsales;
region_group = GROUP region_cust_sales BY region;
region_top2 = FOREACH region_group {
ordersales = ORDER region_cust_sales BY totalsales;
top2 = LIMIT ordersales 2;
GENERATE top2;
};
region_top2 = FOREACH region_top2 GENERATE FLATTEN(top2);
dump region_top2;
```

20. **Batch Processing:**

```
pig retail_top2.pig
```

21. **What Next?**

http://pig.apache.org/docs/r0.15.0/