Professor Satyendra Rana

Big Data Fundamentals CSC6991

Recommendation Engine Implementation Using Pig, Hive, ECL and AWS(Pig)

Final Project Report

Iyad Kuwatly

Summer 2014

Table of Contents

Executive Summary

This report illustrates the usage of Pig, Hive and ECL languages to implement recommendation engine. Hadoop Hortonworks sandbox, HPCC sandbox and Amazon web services were utilized in the implementation. The idea is to present recommendations for investors after analyzing their trades during a period of time and collecting their rating for their experience about their trades during that period. The trade rating is an investor preference that range from 0 to 5 and is based on the overall experience of the investor, that could capital gain, dividends, and any other factor the investor seems important. Usually this data is available for brokers who do the book keeping for all their customers. After processing the data, the system would recommend stocks to trade based on the rating of other investors with similar portfolio (collection of stock trades).

The paper starts with an introduction about the main ideas utilized and a summary the three code scripts (Pig, Hive and ECL) used in the implementation. Then each implementation is illustrated in details with screenshots and output captures in a step by step approach.

Introduction

The input data can be found on stock brokers servers but it is not publically available to protect the privacy of their customers (investors). For illustration, a small data set was manually created and in a step by step it was monitored after each query statement that was applied. At the end of the report, a large data set of 100,000 of random trades values was tested on AWS.

The sample data set has the following items (user vectors):

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

invester_id: identifies a unique investor (the details of the investor are placed in a different table, omitted because it is not relevant to this implementation)

trade_id: identifies a unique trade (the details of the trade including volume and price are placed in a different table; omitted because it is not relevant to this implementation)

rating: indicate the investor overall experience with this trade

In each implementation that is presented afterwards, the input data set is going to be processed in four major steps (**Pseudo code**) as follow:

1-Loading the user vectors to the database [shortcut LOADING]

2- Generating Co-occurrence matrix [shortcut CO-OCCURRENCE]

3- Multiplying Co-occurrence matrix by user vector (user 3 is chosen for illustration) [shortcut MULTIPLICATION]

4- Generating the recommendations properly ranked [shortcut RECOMMENDATIONS]

In the following four pages, full code summary for hive, pig and ECL is as presented followed by detailed discussion for each.

Hive Complete Code

```
CREATE TABLE trades (
  investor_id INT ,
  trade_id INT ,
  rating INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE ;
LOAD DATA
  LOCAL INPATH '/root/trades.txt'
  OVERWRITE INTO TABLE trades;

CREATE TABLE cooccurrence AS
SELECT
  a.trade_id ,
  b.trade_id AS trade_id_2 ,
  COUNT(*) AS tradescount
FROM
  trades a
  JOIN trades b ON a.investor_id = b.investor_id
GROUP BY
  a.trade_id ,
  b.trade_id;
CREATE TABLE product_matrix AS
SELECT
  a.trade_id ,
  a.trade_id_2 ,
 sum(b.rating*a.tradescount) as cooccurrencecount
FROM
cooccurrence a,
trades b
where
a.trade_id_2= b.trade_id
AND b.investor_id=3
group by
a.trade_id,
a.trade_id_2;
CREATE TABLE result_Vector AS
select trade_id , sum(cooccurrencecount) as recommended
from product_matrix
group by trade_id
order by recommended DESC;
```

Pig Complete Code

```
trades = LOAD '/root/trades.txt' USING PigStorage('\t') AS
(investor_id:int, trade_id:int, rating:int) ;

trades_2 = FOREACH trades GENERATE investor_id AS
investor_id_2, trade_id AS trade_id_2, rating AS rating_2 ;
joinedtradeds = JOIN trades BY investor_id, trades_2 BY
investor_id_2 ;
groupedtrades = group  joinedtradeds  by
(trade_id,trade_id_2);
cooccurrence = FOREACH groupedtrades GENERATE
        group.trade_id as trade_id,
        group.trade_id_2 as trade_id_2,
        COUNT($1) as tradecount;
filteredtradesforuser3 = filter trades BY investor_id == 3;
pre_product_matrix = JOIN cooccurrence BY trade_id_2,
filteredtradesforuser3 BY trade_id ;
product_matrix = FOREACH pre_product_matrix GENERATE
        $0 as trade_id,
        $1 as trade_id_2,
        (int)$2*$5 as user3ratingproduct;
grouped_product_matrix = group  product_matrix  by trade_id;
result_Vector = FOREACH grouped_product_matrix GENERATE
        $0 as trade_id,
        SUM(product_matrix.user3ratingproduct) as
user3ratingtotal;
joinedrecommendations = JOIN result_Vector by trade_id LEFT,
filteredtradesforuser3 BY trade_id;
filteredrecommendations = filter joinedrecommendations BY $2
is null;

user3recommendation = FOREACH filteredrecommendations
GENERATE
$0 as trade_id,
$1 as recommendation;

user3recommendationsorted = order user3recommendation by
recommendation desc;
```

ECL Complete Code

```
Layout_TradeData := RECORD
INTEGER4 investor_id;
INTEGER4 trade_id;
INTEGER4 rating;
END;

EXPORT trade := DATASET('~online::ik::project::trades',
            Layout_TradeData, CSV(SEPARATOR('\t'),QUOTE('')));
import $;
Output($.trade, NAMED('Trades'));

Joined_Record := RECORD
    INTEGER4 trade_id;
    INTEGER4 trade_id_2;
END;

Joined_Record JoinThem($.trade L, $.trade R) := TRANSFORM
    SELF.trade_id := L.trade_id;
    SELF.trade_id_2 := R.trade_id;
END;

JoinedTrades := JOIN($.trade,$.trade,
                                LEFT.investor_id = RIGHT.investor_id,
                                              JoinThem(LEFT,RIGHT));

OUTPUT (JoinedTrades,NAMED('JoinedTrades'));

COOCCURRENCE_Record := RECORD
    JoinedTrades.trade_id;
    JoinedTrades.trade_id_2;
    tradescount :=COUNT(GROUP);
END;


COOCCURRENCE_Matrix:=TABLE(JoinedTrades,COOCCURRENCE_Record,JoinedTrades.trad
e_id,JoinedTrades.trade_id_2);
output (COOCCURRENCE_Matrix,NAMED('COOCCURRENCE_Matrix'));

filteredtradesforuser3 := $.trade(investor_id = 3);
output (filteredtradesforuser3,NAMED('filteredtradesforuser3'));

product_Record := RECORD
    COOCCURRENCE_Record.trade_id;
    COOCCURRENCE_Record.trade_id_2;
    INTEGER4 user3ratingproduct;
END;

product_Record MutiplyThem(COOCCURRENCE_Matrix L, filteredtradesforuser3 R)
:= TRANSFORM
    SELF.trade_id:= L.trade_id;
    SELF.trade_id_2:= L.trade_id_2;
    SELF.user3ratingproduct := L.tradescount * R.rating;
END;

product_matrix   := JOIN(COOCCURRENCE_Matrix,filteredtradesforuser3,
```

```
                                    LEFT.trade_id_2 = RIGHT.trade_id,
                                                  MutiplyThem(LEFT,RIGHT));
OUTPUT (product_matrix,NAMED('product_matrix'));

result_Vector := RECORD
    product_matrix.trade_id;
    INTEGER4 user3ratingtotal :=
SUM(GROUP,product_matrix.user3ratingproduct);
END;


recommendations := TABLE(product_matrix,result_Vector,trade_id);
OUTPUT (recommendations,NAMED('recommendations'));

filtered_result_record := RECORD
    INTEGER4 trade_id;
    INTEGER4 user3ratingtotal;
END;

filtered_result_record JoinRecs(result_Vector L, filteredtradesforuser3 R) :=
TRANSFORM
    SELF.trade_id := L.trade_id;
    SELF.user3ratingtotal := L.user3ratingtotal;
END;

filtered_result := JOIN(recommendations,filteredtradesforuser3,
                                    LEFT.trade_id = RIGHT.trade_id,
                                                  JoinRecs(LEFT,RIGHT),
                                    LEFT ONLY);
OUTPUT (filtered_result,NAMED('filtered_result'));

sorted_filtered_result := Sort(filtered_result,-user3ratingtotal);
OUTPUT (sorted_filtered_result,NAMED('sorted_filtered_result'));
```

Part1.1 Hive LOADING

Before (Trades.txt on notepad)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

After (trades table on hive)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

Code

```
CREATE TABLE trades (
  investor_id INT ,
  trade_id INT ,
  rating INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE ;
LOAD DATA
  LOCAL INPATH '/root/trades.txt'
  OVERWRITE INTO TABLE trades;
```

```
[root@sandbox ~]# hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
hive> CREATE TABLE trades (
    >   investor_id INT ,
    >   trade_id INT ,
    >   rating INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '\t'
    > STORED AS TEXTFILE ;
OK
Time taken: 2.62 seconds
hive> SHOW TABLES;
OK
sample_07
sample_08
trades
Time taken: 0.607 seconds, Fetched: 3 row(s)
hive> DESCRIBE trades;
OK
investor_id             int
trade_id                int
rating                  int
Time taken: 0.566 seconds, Fetched: 3 row(s)
hive> LOAD DATA
    >   LOCAL INPATH '/root/trades.txt'
    >   OVERWRITE INTO TABLE trades;
Copying data from file:/root/trades.txt
Copying file: file:/root/trades.txt
Loading data to table default.trades
rmr: DEPRECATED: Please use 'rm -r' instead.
Moved: 'hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/trades' to trash at:
hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
Table default.trades stats: [numFiles=1, numRows=0, totalSize=205, rawDataSize=0]
OK
Time taken: 1.298 seconds
hive> select * from trades;
OK
1       101     1
2       101     2
3       101     3
4       101     4
5       101     5
6       101     5
2       102     1
3       102     2
5       102     3
1       103     1
2       103     2
```

```
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1
Time taken: 0.864 seconds, Fetched: 23 row(s)
hive>
```

Part1.2 Hive CO-OCCURRENCE

Before (trades table)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

After (cooccurrence table)

	101	102	103	104	105	106	107
101	6	3	4	4	3	2	1
102	3	3	3	2	1	1	
103	4	3	4	3	1	2	
104	4	2	3	4	2	2	1
105	3	1	1	2	3	1	1
106	2	1	2	2	1	2	
107	1			1	1		1

Code

```
CREATE TABLE cooccurrence AS
SELECT
  a.trade_id ,
  b.trade_id AS trade_id_2 ,
  COUNT(*) AS tradescount
FROM
  trades a
  JOIN trades b ON a.investor_id = b.investor_id
GROUP BY
  a.trade_id ,
  b.trade_id;
```

```
hive> CREATE TABLE cooccurrence AS
    > SELECT
    >   a.trade_id ,
    >   b.trade_id AS trade_id_2 ,
    >   COUNT(*) AS tradescount
    > FROM
    >   trades a
    >   JOIN trades b ON a.investor_id = b.investor_id
    > GROUP BY
    >   a.trade_id ,
    >   b.trade_id;
Query ID = root_20140727073636_d9567595-8745-4871-b783-b3ec45747297
Total jobs = 1
14/07/27 07:36:47 WARN conf.Configuration: file:/tmp/root/hive_2014-07-27_07-36-
43_086_4055741953442478727-1/-local-10007/jobconf.xml:an attempt to override final parameter:
mapreduce.job.end-notification.max.retry.interval;  Ignoring.
14/07/27 07:36:47 WARN conf.Configuration: file:/tmp/root/hive_2014-07-27_07-36-
43_086_4055741953442478727-1/-local-10007/jobconf.xml:an attempt to override final parameter:
mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/root/root_20140727073636_d9567595-8745-4871-b783-b3ec45747297.log
2014-07-27 07:36:48     Starting to launch local task to process map join;      maximum memory =
260177920
2014-07-27 07:36:49     Dump the side-table into file: file:/tmp/root/hive_2014-07-27_07-36-
43_086_4055741953442478727-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile00--.hashtable
2014-07-27 07:36:49     Uploaded 1 File to: file:/tmp/root/hive_2014-07-27_07-36-
43_086_4055741953442478727-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile00--.hashtable (482
bytes)
2014-07-27 07:36:49     End of local task; Time Taken: 1.045 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 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_1406464910035_0008, Tracking URL =
http://sandbox.hortonworks.com:8088/proxy/application_1406464910035_0008/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1406464910035_0008
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2014-07-27 07:37:00,927 Stage-2 map = 0%,  reduce = 0%
2014-07-27 07:37:06,608 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.24 sec
2014-07-27 07:37:14,343 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.86 sec
MapReduce Total cumulative CPU time: 2 seconds 860 msec
Ended Job = job_1406464910035_0008
Moving data to: hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/cooccurrence
```

```
Table default.cooccurrence stats: [numFiles=1, numRows=43, totalSize=430, rawDataSize=387]
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.86 sec   HDFS Read: 432 HDFS Write: 507 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 860 msec
OK
Time taken: 32.728 seconds
hive> select * from cooccurrence;
OK
101	101	6
101	102	3
101	103	4
101	104	4
101	105	3
101	106	2
101	107	1
102	101	3
102	102	3
102	103	3
102	104	2
102	105	1
102	106	1
103	101	4
103	102	3
103	103	4
103	104	3
103	105	1
103	106	2
104	101	4
104	102	2
104	103	3
104	104	4
104	105	2
104	106	2
104	107	1
105	101	3
105	102	1
105	103	1
105	104	2
105	105	3
105	106	1
105	107	1
106	101	2
106	102	1
106	103	2
106	104	2
106	105	1
106	106	2
107	101	1
```

```
107     104     1
107     105     1
107     107     1
Time taken: 0.372 seconds, Fetched: 43 row(s)
hive>
```

Part1.3 Hive MULTIPLICATION Part 1

Before (cooccurrence table)

	101	102	103	104	105	106	107
101	6	3	4	4	3	2	1
102	3	3	3	2	1	1	
103	4	3	4	3	1	2	
104	4	2	3	4	2	2	1
105	3	1	1	2	3	1	1
106	2	1	2	2	1	2	
107	1			1	1		1

Before(user 3 vector)

101	3
102	2
103	3
104	2
105	0
106	0
107	0

After (product_matrix table)

	101	102	103	104	105	106	107
101	18	6	12	8	0	0	0
102	9	6	9	4	0	0	0
103	12	6	12	6	0	0	0
104	12	4	9	8	0	0	0
105	9	2	3	4	0	0	0
106	6	2	6	4	0	0	0
107	3	0	0	2	0	0	0

Code

```
CREATE TABLE product_matrix AS
SELECT
  a.trade_id ,
  a.trade_id_2 ,
 sum(b.rating*a.tradescount) as cooccurrencecount
FROM
cooccurrence a,
trades b
where
a.trade_id_2= b.trade_id
AND b.investor_id=3
group by
a.trade_id,
a.trade_id_2;
```

```
hive> CREATE TABLE product_matrix AS
    > SELECT
    >   a.trade_id ,
    >   a.trade_id_2 ,
    >  sum(b.rating*a.tradescount) as cooccurrencecount
    > FROM
    > cooccurrence a,
    > trades b
    > where
    > a.trade_id_2= b.trade_id
    > AND b.investor_id=3
    > group by
    > a.trade_id,
    > a.trade_id_2;
Query ID = root_20140727073838_4d464e0d-e201-4538-8ae7-2f42c1effeb9
Total jobs = 1
14/07/27 07:38:07 WARN conf.Configuration: file:/tmp/root/hive_2014-07-27_07-38-
03_834_2227333854603118385-1/-local-10007/jobconf.xml:an attempt to override final parameter:
mapreduce.job.end-notification.max.retry.interval;  Ignoring.
14/07/27 07:38:07 WARN conf.Configuration: file:/tmp/root/hive_2014-07-27_07-38-
03_834_2227333854603118385-1/-local-10007/jobconf.xml:an attempt to override final parameter:
mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/root/root_20140727073838_4d464e0d-e201-4538-8ae7-2f42c1effeb9.log
2014-07-27 07:38:08     Starting to launch local task to process map join;      maximum memory =
260177920
2014-07-27 07:38:10     Dump the side-table into file: file:/tmp/root/hive_2014-07-27_07-38-
03_834_2227333854603118385-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile11--.hashtable
2014-07-27 07:38:10     Uploaded 1 File to: file:/tmp/root/hive_2014-07-27_07-38-
03_834_2227333854603118385-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile11--.hashtable (348
bytes)
2014-07-27 07:38:10     End of local task; Time Taken: 1.457 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 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_1406464910035_0009, Tracking URL =
http://sandbox.hortonworks.com:8088/proxy/application_1406464910035_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1406464910035_0009
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2014-07-27 07:38:18,814 Stage-2 map = 0%,  reduce = 0%
2014-07-27 07:38:25,219 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.8 sec
2014-07-27 07:38:34,040 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.7 sec
```

```
MapReduce Total cumulative CPU time: 3 seconds 700 msec
Ended Job = job_1406464910035_0009
Moving data to: hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/product_matrix
Table default.product_matrix stats: [numFiles=1, numRows=26, totalSize=265, rawDataSize=239]
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 3.7 sec   HDFS Read: 661 HDFS Write: 344 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 700 msec
OK
Time taken: 31.622 seconds
hive> select * from product_matrix;
OK
101     101     18
101     102     6
101     103     12
101     104     8
102     101     9
102     102     6
102     103     9
102     104     4
103     101     12
103     102     6
103     103     12
103     104     6
104     101     12
104     102     4
104     103     9
104     104     8
105     101     9
105     102     2
105     103     3
105     104     4
106     101     6
106     102     2
106     103     6
106     104     4
107     101     3
107     104     2
Time taken: 0.334 seconds, Fetched: 26 row(s)
```

Part1.3 Hive MULTIPLICATION Part 2

Before (product_matrix table)

	101	102	103	104	105	106	107
101	18	6	12	8	0	0	0
102	9	6	9	4	0	0	0
103	12	6	12	6	0	0	0
104	12	4	9	8	0	0	0
105	9	2	3	4	0	0	0
106	6	2	6	4	0	0	0
107	3	0	0	2	0	0	0

After (result_Vector table)

101	44
102	28
103	36
104	33
105	18
106	18
107	5

Code

```
CREATE TABLE result_Vector AS
select trade_id , sum(cooccurrencecount) as recommended
from product_matrix
group by trade_id
order by recommended DESC
```

```
hive> CREATE TABLE result_Vector AS
    > select trade_id , sum(cooccurrencecount) as recommended
    > from product_matrix
    > group by trade_id
    > order by recommended DESC;
Query ID = root_20140727073939_49affff7-d9cc-40ee-9ba2-06b0eefb650d
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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_1406464910035_0010, Tracking URL =
http://sandbox.hortonworks.com:8088/proxy/application_1406464910035_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1406464910035_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2014-07-27 07:39:15,354 Stage-1 map = 0%,  reduce = 0%
2014-07-27 07:39:21,922 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.35 sec
2014-07-27 07:39:28,464 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.69 sec
MapReduce Total cumulative CPU time: 2 seconds 690 msec
Ended Job = job_1406464910035_0010
Launching Job 2 out of 2
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_1406464910035_0011, Tracking URL =
http://sandbox.hortonworks.com:8088/proxy/application_1406464910035_0011/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1406464910035_0011
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2014-07-27 07:39:36,881 Stage-2 map = 0%,  reduce = 0%
2014-07-27 07:39:43,283 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.13 sec
2014-07-27 07:39:49,755 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.53 sec
MapReduce Total cumulative CPU time: 2 seconds 530 msec
Ended Job = job_1406464910035_0011
Moving data to: hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/result_vector
Table default.result_vector stats: [numFiles=1, numRows=0, totalSize=48, rawDataSize=0]
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.69 sec   HDFS Read: 498 HDFS Write: 229 SUCCESS
Job 1: Map: 1  Reduce: 1   Cumulative CPU: 2.53 sec   HDFS Read: 606 HDFS Write: 125 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 220 msec
OK
```

```
Time taken: 43.36 seconds
hive> select * from result_Vector;
OK
101	44
103	36
104	33
102	28
106	18
105	18
107	5
Time taken: 0.356 seconds, Fetched: 7 row(s)
```

Part1.4 Hive RECOMMENDATIONS

Before (result_Vector table)

101	44
102	28
103	36
104	33
105	18
106	18
107	5

After (User3_recommendations table)

105	18
106	18
107	5

Code

```
CREATE TABLE User3_recommendations AS
SELECT  DISTINCT a.trade_id, a.recommended
FROM    result_vector a
        LEFT JOIN trades b
          ON a.trade_id = b.trade_id AND b.investor_id = 3
WHERE   b.rating IS NULL
order by a.recommended DESC;
```

```
hive> CREATE TABLE User3_recommendations AS
    > SELECT  DISTINCT a.trade_id, a.recommended
    > FROM    result_vector a
    >         LEFT JOIN trades b
    >           ON a.trade_id = b.trade_id AND b.investor_id = 3
    > WHERE   b.rating IS NULL
    > ORDER BY a.recommended DESC;
Query ID = root_20140727074040_b89d6d25-1b94-4ee2-a0b7-973751b1ab73
Total jobs = 1
14/07/27 07:40:23 WARN conf.Configuration: file:/tmp/root/hive_2014-07-27_07-40-
17_726_7960976239964976414-1/-local-10007/jobconf.xml:an attempt to override final parameter:
mapreduce.job.end-notification.max.retry.interval;  Ignoring.
14/07/27 07:40:23 WARN conf.Configuration: file:/tmp/root/hive_2014-07-27_07-40-
17_726_7960976239964976414-1/-local-10007/jobconf.xml:an attempt to override final parameter:
mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/root/root_20140727074040_b89d6d25-1b94-4ee2-a0b7-973751b1ab73.log
2014-07-27 07:40:24     Starting to launch local task to process map join;      maximum memory =
260177920
2014-07-27 07:40:26     Dump the side-table into file: file:/tmp/root/hive_2014-07-27_07-40-
17_726_7960976239964976414-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile21--.hashtable
2014-07-27 07:40:26     Uploaded 1 File to: file:/tmp/root/hive_2014-07-27_07-40-
17_726_7960976239964976414-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile21--.hashtable (340
bytes)
2014-07-27 07:40:26     End of local task; Time Taken: 2.128 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 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_1406464910035_0012, Tracking URL =
http://sandbox.hortonworks.com:8088/proxy/application_1406464910035_0012/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1406464910035_0012
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2014-07-27 07:40:35,721 Stage-2 map = 0%,  reduce = 0%
2014-07-27 07:40:42,213 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.69 sec
2014-07-27 07:40:48,624 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.23 sec
MapReduce Total cumulative CPU time: 3 seconds 230 msec
Ended Job = job_1406464910035_0012
Moving data to: hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/user3_recommendations
Table default.user3_recommendations stats: [numFiles=1, numRows=0, totalSize=20, rawDataSize=0]
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 3.23 sec   HDFS Read: 280 HDFS Write: 105 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 230 msec
```

```
OK
Time taken: 32.271 seconds


hive> select * from User3_recommendations;
OK
105     18
106     18
107     5
Time taken: 0.294 seconds, Fetched: 3 row(s)
hive>
```

Part2.1 Pig LOADING

Before (Trades.txt on notepad)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

After (trades table on pig)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

Code

```
trades = LOAD '/root/trades.txt' USING PigStorage('\t')
AS (investor_id:int, trade_id:int, rating:int) ;
```

```
[root@sandbox ~]# pig -x local
2014-07-27 15:29:49,852 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.1.2.1.1.0-385
(rexported) compiled Apr 16 2014, 15:59:00
2014-07-27 15:29:49,853 [main] INFO  org.apache.pig.Main - Logging error messages to:
/root/pig_1406500189851.log
2014-07-27 15:29:49,874 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file
/root/.pigbootup not found
2014-07-27 15:29:50,102 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2014-07-27 15:29:50,102 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2014-07-27 15:29:50,104 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system
at: file:///
2014-07-27 15:29:50,511 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2014-07-27 15:29:50,514 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> trades = LOAD '/root/trades.txt' USING PigStorage('\t') AS (investor_id:int, trade_id:int, rating:int)
;
grunt>
grunt> describe trades;
trades: {investor_id: int,trade_id: int,rating: int}
grunt> illustrate trades;
-------------------------------------------------------------------
| trades     | investor_id:int   | trade_id:int   | rating:int   |
-------------------------------------------------------------------
|            | 5                 | 103            | 4            |
-------------------------------------------------------------------
grunt> dump trades;
(1,101,1)
(2,101,2)
(3,101,3)
(4,101,4)
(5,101,5)
(6,101,5)
(2,102,1)
(3,102,2)
(5,102,3)
(1,103,1)
(2,103,2)
(3,103,3)
(5,103,4)
(1,104,1)
(3,104,2)
(5,104,4)
(6,104,5)
(4,105,1)
```

```
(5,105,2)
(6,105,5)
(1,106,1)
(5,106,2)
(6,107,1)
```

Part2.2 Pig CO-OCCURRENCE

Before (trades table)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

After (cooccurrence table)

	101	102	103	104	105	106	107
101	6	3	4	4	3	2	1
102	3	3	3	2	1	1	
103	4	3	4	3	1	2	
104	4	2	3	4	2	2	1
105	3	1	1	2	3	1	1
106	2	1	2	2	1	2	
107	1			1	1		1

Code

```
trades_2 = FOREACH trades GENERATE investor_id AS
investor_id_2, trade_id AS trade_id_2, rating AS rating_2 ;
joinedtradeds = JOIN trades BY investor_id, trades_2 BY
investor_id_2 ;
groupedtrades = group  joinedtradeds  by
(trade_id,trade_id_2);
cooccurrence = FOREACH groupedtrades GENERATE
        group.trade_id as trade_id,
        group.trade_id_2 as trade_id_2,
        COUNT($1) as tradecount;
```

```
grunt> trades_2 = FOREACH trades GENERATE investor_id AS investor_id_2, trade_id AS trade_id_2,
rating AS rating_2 ;
grunt> joinedtradeds = JOIN trades BY investor_id, trades_2 BY investor_id_2 ;
grunt> groupedtrades = group  joinedtradeds  by (trade_id,trade_id_2);
grunt> cooccurrence = FOREACH groupedtrades GENERATE
>> group.trade_id as trade_id,
>> group.trade_id_2 as trade_id_2,
>> COUNT($1) as tradecount;
grunt>
grunt> describe trades_2;
trades_2: {investor_id_2: int,trade_id_2: int,rating_2: int}
grunt> dump trades_2;
(1,101,1)
(2,101,2)
(3,101,3)
(4,101,4)
(5,101,5)
(6,101,5)
(2,102,1)
(3,102,2)
(5,102,3)
(1,103,1)
(2,103,2)
(3,103,3)
(5,103,4)
(1,104,1)
(3,104,2)
(5,104,4)
(6,104,5)
(4,105,1)
(5,105,2)
(6,105,5)
(1,106,1)
(5,106,2)
(6,107,1)

grunt> describe joinedtradeds;
joinedtradeds: {trades::investor_id: int,trades::trade_id: int,trades::rating: int,trades_2::investor_id_2:
int,trades_2::trade_id_2: int,trades_2::rating_2: int}
grunt> dump trades_2;
(1,104,1,1,104,1)
(1,104,1,1,106,1)
(1,104,1,1,103,1)
(1,104,1,1,101,1)
(1,106,1,1,104,1)
(1,106,1,1,106,1)
(1,106,1,1,103,1)
(1,106,1,1,101,1)
```

```
(1,103,1,1,104,1)
(1,103,1,1,106,1)
(1,103,1,1,103,1)
(1,103,1,1,101,1)
(1,101,1,1,104,1)
(1,101,1,1,106,1)
(1,101,1,1,103,1)
(1,101,1,1,101,1)
(2,101,2,2,101,2)
(2,101,2,2,102,1)
(2,101,2,2,103,2)
(2,102,1,2,101,2)
(2,102,1,2,102,1)
(2,102,1,2,103,2)
(2,103,2,2,101,2)
(2,103,2,2,102,1)
(2,103,2,2,103,2)
(3,103,3,3,103,3)
(3,103,3,3,104,2)
(3,103,3,3,102,2)
(3,103,3,3,101,3)
(3,104,2,3,103,3)
(3,104,2,3,104,2)
(3,104,2,3,102,2)
(3,104,2,3,101,3)
(3,102,2,3,103,3)
(3,102,2,3,104,2)
(3,102,2,3,102,2)
(3,102,2,3,101,3)
(3,101,3,3,103,3)
(3,101,3,3,104,2)
(3,101,3,3,102,2)
(3,101,3,3,101,3)
(4,105,1,4,105,1)
(4,105,1,4,101,4)
(4,101,4,4,105,1)
(4,101,4,4,101,4)
(5,105,2,5,105,2)
(5,105,2,5,102,3)
(5,105,2,5,106,2)
(5,105,2,5,101,5)
(5,105,2,5,104,4)
(5,105,2,5,103,4)
(5,102,3,5,105,2)
(5,102,3,5,102,3)
(5,102,3,5,106,2)
(5,102,3,5,101,5)
(5,102,3,5,104,4)
```

```
(5,102,3,5,103,4)
(5,106,2,5,105,2)
(5,106,2,5,102,3)
(5,106,2,5,106,2)
(5,106,2,5,101,5)
(5,106,2,5,104,4)
(5,106,2,5,103,4)
(5,101,5,5,105,2)
(5,101,5,5,102,3)
(5,101,5,5,106,2)
(5,101,5,5,101,5)
(5,101,5,5,104,4)
(5,101,5,5,103,4)
(5,104,4,5,105,2)
(5,104,4,5,102,3)
(5,104,4,5,106,2)
(5,104,4,5,101,5)
(5,104,4,5,104,4)
(5,104,4,5,103,4)
(5,103,4,5,105,2)
(5,103,4,5,102,3)
(5,103,4,5,106,2)
(5,103,4,5,101,5)
(5,103,4,5,104,4)
(5,103,4,5,103,4)
(6,104,5,6,104,5)
(6,104,5,6,105,5)
(6,104,5,6,101,5)
(6,104,5,6,107,1)
(6,105,5,6,104,5)
(6,105,5,6,105,5)
(6,105,5,6,101,5)
(6,105,5,6,107,1)
(6,101,5,6,104,5)
(6,101,5,6,105,5)
(6,101,5,6,101,5)
(6,101,5,6,107,1)
(6,107,1,6,104,5)
(6,107,1,6,105,5)
(6,107,1,6,101,5)
(6,107,1,6,107,1)


grunt> describe groupedtrades;
groupedtrades: {group: (trades::trade_id: int,trades_2::trade_id_2: int),joinedtradeds:
{(trades::investor_id: int,trades::trade_id: int,trades::rating: int,trades_2::investor_id_2:
int,trades_2::trade_id_2: int,trades_2::rating_2: int)}}
grunt> dump groupedtrades;
```

```
((101,101),{(2,101,2,2,101,2),(5,101,5,5,101,5),(4,101,4,4,101,4),(1,101,1,1,101,1),(3,101,3,3,101,3),(6,1
01,5,6,101,5)})
((101,102),{(3,101,3,3,102,2),(2,101,2,2,102,1),(5,101,5,5,102,3)})
((101,103),{(1,101,1,1,103,1),(5,101,5,5,103,4),(3,101,3,3,103,3),(2,101,2,2,103,2)})
((101,104),{(1,101,1,1,104,1),(6,101,5,6,104,5),(3,101,3,3,104,2),(5,101,5,5,104,4)})
((101,105),{(6,101,5,6,105,5),(5,101,5,5,105,2),(4,101,4,4,105,1)})
((101,106),{(5,101,5,5,106,2),(1,101,1,1,106,1)})
((101,107),{(6,101,5,6,107,1)})
((102,101),{(3,102,2,3,101,3),(5,102,3,5,101,5),(2,102,1,2,101,2)})
((102,102),{(3,102,2,3,102,2),(5,102,3,5,102,3),(2,102,1,2,102,1)})
((102,103),{(2,102,1,2,103,2),(3,102,2,3,103,3),(5,102,3,5,103,4)})
((102,104),{(5,102,3,5,104,4),(3,102,2,3,104,2)})
((102,105),{(5,102,3,5,105,2)})
((102,106),{(5,102,3,5,106,2)})
((103,101),{(1,103,1,1,101,1),(5,103,4,5,101,5),(3,103,3,3,101,3),(2,103,2,2,101,2)})
((103,102),{(5,103,4,5,102,3),(3,103,3,3,102,2),(2,103,2,2,102,1)})
((103,103),{(3,103,3,3,103,3),(2,103,2,2,103,2),(1,103,1,1,103,1),(5,103,4,5,103,4)})
((103,104),{(5,103,4,5,104,4),(3,103,3,3,104,2),(1,103,1,1,104,1)})
((103,105),{(5,103,4,5,105,2)})
((103,106),{(1,103,1,1,106,1),(5,103,4,5,106,2)})
((104,101),{(3,104,2,3,101,3),(1,104,1,1,101,1),(5,104,4,5,101,5),(6,104,5,6,101,5)})
((104,102),{(3,104,2,3,102,2),(5,104,4,5,102,3)})
((104,103),{(1,104,1,1,103,1),(5,104,4,5,103,4),(3,104,2,3,103,3)})
((104,104),{(6,104,5,6,104,5),(5,104,4,5,104,4),(1,104,1,1,104,1),(3,104,2,3,104,2)})
((104,105),{(5,104,4,5,105,2),(6,104,5,6,105,5)})
((104,106),{(5,104,4,5,106,2),(1,104,1,1,106,1)})
((104,107),{(6,104,5,6,107,1)})
((105,101),{(5,105,2,5,101,5),(6,105,5,6,101,5),(4,105,1,4,101,4)})
((105,102),{(5,105,2,5,102,3)})
((105,103),{(5,105,2,5,103,4)})
((105,104),{(6,105,5,6,104,5),(5,105,2,5,104,4)})
((105,105),{(5,105,2,5,105,2),(6,105,5,6,105,5),(4,105,1,4,105,1)})
((105,106),{(5,105,2,5,106,2)})
((105,107),{(6,105,5,6,107,1)})
((106,101),{(1,106,1,1,101,1),(5,106,2,5,101,5)})
((106,102),{(5,106,2,5,102,3)})
((106,103),{(5,106,2,5,103,4),(1,106,1,1,103,1)})
((106,104),{(1,106,1,1,104,1),(5,106,2,5,104,4)})
((106,105),{(5,106,2,5,105,2)})
((106,106),{(5,106,2,5,106,2),(1,106,1,1,106,1)})
((107,101),{(6,107,1,6,101,5)})
((107,104),{(6,107,1,6,104,5)})
((107,105),{(6,107,1,6,105,5)})
((107,107),{(6,107,1,6,107,1)})

grunt> describe cooccurrence;
cooccurrence: {trade_id: int,trade_id_2: int,tradecount: long}
grunt> dump cooccurrence;
```

```
(101,101,6)
(101,102,3)
(101,103,4)
(101,104,4)
(101,105,3)
(101,106,2)
(101,107,1)
(102,101,3)
(102,102,3)
(102,103,3)
(102,104,2)
(102,105,1)
(102,106,1)
(103,101,4)
(103,102,3)
(103,103,4)
(103,104,3)
(103,105,1)
(103,106,2)
(104,101,4)
(104,102,2)
(104,103,3)
(104,104,4)
(104,105,2)
(104,106,2)
(104,107,1)
(105,101,3)
(105,102,1)
(105,103,1)
(105,104,2)
(105,105,3)
(105,106,1)
(105,107,1)
(106,101,2)
(106,102,1)
(106,103,2)
(106,104,2)
(106,105,1)
(106,106,2)
(107,101,1)
(107,104,1)
(107,105,1)
(107,107,1)
```

Part2.3 Pig MULTIPLICATION Part 1

Before (cooccurrence table)

	101	102	103	104	105	106	107
101	6	3	4	4	3	2	1
102	3	3	3	2	1	1	
103	4	3	4	3	1	2	
104	4	2	3	4	2	2	1
105	3	1	1	2	3	1	1
106	2	1	2	2	1	2	
107	1			1	1		1

Before(user 3 vector)

101	3
102	2
103	3
104	2
105	0
106	0
107	0

After (product_matrix table)

	101	102	103	104	105	106	107
101	18	6	12	8	0	0	0
102	9	6	9	4	0	0	0
103	12	6	12	6	0	0	0
104	12	4	9	8	0	0	0
105	9	2	3	4	0	0	0
106	6	2	6	4	0	0	0
107	3	0	0	2	0	0	0

Code

```
filteredtradesforuser3 = filter trades BY investor_id == 3;
pre_product_matrix = JOIN cooccurrence BY trade_id_2,
filteredtradesforuser3 BY trade_id ;
product_matrix = FOREACH pre_product_matrix GENERATE
        $0 as trade_id,
        $1 as trade_id_2,
        (int)$2*$5 as user3ratingproduct;
grouped_product_matrix = group  product_matrix  by trade_id;
```

```
grunt> filteredtradesforuser3 = filter trades BY investor_id == 3;
grunt> pre_product_matrix = JOIN cooccurrence BY trade_id_2, filteredtradesforuser3 BY trade_id ;
grunt> product_matrix = FOREACH pre_product_matrix GENERATE
>> $0 as trade_id,
>> $1 as trade_id_2,
>> (int)$2*$5 as user3ratingproduct;
grunt> grouped_product_matrix = group  product_matrix  by trade_id;
grunt>



grunt> describe filteredtradesforuser3;
filteredtradesforuser3: {investor_id: int,trade_id: int,rating: int}
grunt> dump filteredtradesforuser3;
(3,101,3)
(3,102,2)
(3,103,3)
(3,104,2)


grunt> describe pre_product_matrix;
pre_product_matrix: {cooccurrence::trade_id: int,cooccurrence::trade_id_2:
int,cooccurrence::tradecount: long,filteredtradesforuser3::investor_id:
int,filteredtradesforuser3::trade_id: int,filteredtradesforuser3::rating: int}
grunt> dump pre_product_matrix;
(101,101,6,3,101,3)
(102,101,3,3,101,3)
(107,101,1,3,101,3)
(103,101,4,3,101,3)
(105,101,3,3,101,3)
(106,101,2,3,101,3)
(104,101,4,3,101,3)
(101,102,3,3,102,2)
(105,102,1,3,102,2)
(106,102,1,3,102,2)
(102,102,3,3,102,2)
(104,102,2,3,102,2)
(103,102,3,3,102,2)
(104,103,3,3,103,3)
(106,103,2,3,103,3)
(105,103,1,3,103,3)
(103,103,4,3,103,3)
(102,103,3,3,103,3)
(101,103,4,3,103,3)
(103,104,3,3,104,2)
(102,104,2,3,104,2)
(106,104,2,3,104,2)
(105,104,2,3,104,2)
```

```
(101,104,4,3,104,2)
(107,104,1,3,104,2)
(104,104,4,3,104,2)


grunt> describe product_matrix;
product_matrix: {trade_id: int,trade_id_2: int,user3ratingproduct: int}
grunt> dump product_matrix;
(101,101,18)
(102,101,9)
(107,101,3)
(103,101,12)
(105,101,9)
(106,101,6)
(104,101,12)
(101,102,6)
(105,102,2)
(106,102,2)
(102,102,6)
(104,102,4)
(103,102,6)
(104,103,9)
(106,103,6)
(105,103,3)
(103,103,12)
(102,103,9)
(101,103,12)
(103,104,6)
(102,104,4)
(106,104,4)
(105,104,4)
(101,104,8)
(107,104,2)
(104,104,8)


grunt> describe grouped_product_matrix;
grouped_product_matrix: {group: int,product_matrix: {(trade_id: int,trade_id_2: int,user3ratingproduct:
int)}}
grunt> dump grouped_product_matrix;
(101,{(101,104,8),(101,101,18),(101,102,6),(101,103,12)})
(102,{(102,103,9),(102,102,6),(102,104,4),(102,101,9)})
(103,{(103,102,6),(103,104,6),(103,103,12),(103,101,12)})
(104,{(104,103,9),(104,102,4),(104,101,12),(104,104,8)})
(105,{(105,102,2),(105,104,4),(105,101,9),(105,103,3)})
(106,{(106,102,2),(106,101,6),(106,103,6),(106,104,4)})
(107,{(107,104,2),(107,101,3)})
```

Part2.3 Pig MULTIPLICATION Part 2

Before (product_matrix table)

	101	102	103	104	105	106	107
101	18	6	12	8	0	0	0
102	9	6	9	4	0	0	0
103	12	6	12	6	0	0	0
104	12	4	9	8	0	0	0
105	9	2	3	4	0	0	0
106	6	2	6	4	0	0	0
107	3	0	0	2	0	0	0

After (result_Vector table)

101	44
102	28
103	36
104	33
105	18
106	18
107	5

Code

```
result_Vector = FOREACH grouped_product_matrix GENERATE
        $0 as trade_id,
        SUM(product_matrix.user3ratingproduct) as
user3ratingtotal;
```

```
grunt> result_Vector = FOREACH grouped_product_matrix GENERATE
>> $0 as trade_id,
>> SUM(product_matrix.user3ratingproduct) as user3ratingtotal;
grunt>

grunt> describe result_Vector;
result_Vector: {trade_id: int,user3ratingtotal: long}
grunt> dump result_Vector;
(101,44)
(102,28)
(103,36)
(104,33)
(105,18)
(106,18)
(107,5)
```

Part2.4 Pig RECOMMENDATIONS

Before (result_Vector table)

101	44
102	28
103	36
104	33
105	18
106	18
107	5

After (User3_recommendations table)

105	18
106	18
107	5

Code

```
joinedrecommendations = JOIN result_Vector by trade_id LEFT,
filteredtradesforuser3 BY trade_id;
filteredrecommendations = filter joinedrecommendations BY $2
is null;

user3recommendation = FOREACH filteredrecommendations GENERATE
$0 as trade_id,
$1 as recommendation;

user3recommendationsorted = order user3recommendation by
recommendation desc;
```

```
grunt> joinedrecommendations = JOIN result_Vector by trade_id LEFT, filteredtradesforuser3 BY
trade_id;
grunt> filteredrecommendations = filter joinedrecommendations BY $2 is null;
grunt>
grunt> user3recommendation = FOREACH filteredrecommendations GENERATE
>> $0 as trade_id,
>> $1 as recommendation;
grunt>
grunt> user3recommendationsorted = order user3recommendation by recommendation desc;
grunt>
grunt> describe joinedrecommendations;
joinedrecommendations: {result_Vector::trade_id: int,result_Vector::user3ratingtotal:
long,filteredtradesforuser3::investor_id: int,filteredtradesforuser3::trade_id:
int,filteredtradesforuser3::rating: int}
grunt> dump joinedrecommendations;
(101,44,3,101,3)
(102,28,3,102,2)
(103,36,3,103,3)
(104,33,3,104,2)
(105,18,,,)
(106,18,,,)
(107,5,,,)


grunt> describe filteredrecommendations;
filteredrecommendations: {result_Vector::trade_id: int,result_Vector::user3ratingtotal:
long,filteredtradesforuser3::investor_id: int,filteredtradesforuser3::trade_id:
int,filteredtradesforuser3::rating: int}
grunt> dump filteredrecommendations;
(105,18,,,)
(106,18,,,)
(107,5,,,)

grunt> describe user3recommendation;
user3recommendation: {trade_id: int,recommendation: long}
grunt> dump user3recommendation;
(105,18)
(106,18)
(107,5)

grunt> describe user3recommendationsorted;
user3recommendationsorted: {trade_id: int,recommendation: long}
grunt> dump user3recommendationsorted;
(106,18)
(105,18)
(107,5)
```

Pig ILLUSTRATION

trades	investor_id:int	trade_id:int	rating:int
	5	103	4

trades	investor_id:int	trade_id:int	rating:int
	5	106	2
	5	102	3
	3	104	2
	3	101	3
	5	101	5

trades	investor_id:int	trade_id:int	rating:int

trades	investor_id:int	trade_id:int	rating:int

trades	investor_id:int	trade_id:int	rating:int

trades_2	investor_id_2:int	trade_id_2:int	rating_2:int
	5	106	2
	5	102	3
	3	104	2
	3	101	3
	5	101	5

joinedtradeds	trades::investor_id:int	trades::trade_id:int	trades::rating:int	trades_2::investor_id_2:int	trades_2::trade_id_2:int	trades_2::rating_2:int
	3	104	2	3	104	2
	3	104	2	3	101	3
	3	101	3	3	104	2
	3	101	3	3	101	3
	5	106	2	5	106	2
	5	106	2	5	102	3
	5	106	2	5	101	5
	5	102	3	5	106	2
	5	102	3	5	102	3
	5	102	3	5	101	5
	5	101	5	5	106	2
	5	101	5	5	102	3
	5	101	5	5	101	5

groupedtrades	group:tuple(trades::trade id:int,trades 2::trade id 2:int)	joinedtradeds:bag{:tuple(trades::investor id:int,…::rating 2:int)}
	(101, 101)	{(3, ..., 3), (5, ..., 5)}
	(101, 102)	{(5, ..., 3)}
	(101, 104)	{(3, ..., 2)}
	(101, 106)	{(5, ..., 2)}
	(102, 101)	{(5, ..., 5)}
	(102, 102)	{(5, ..., 3)}
	(102, 106)	{(5, ..., 2)}
	(104, 101)	{(3, ..., 3)}
	(104, 104)	{(3, ..., 2)}
	(106, 101)	{(5, ..., 5)}
	(106, 102)	{(5, ..., 3)}
	(106, 106)	{(5, ..., 2)}

cooccurrence	trade id:int	trade id 2:int	tradecount:long
	101	101	2
	101	102	1
	101	104	1
	101	106	1
	102	101	1
	102	102	1
	102	106	1
	104	101	1
	104	104	1
	106	101	1
	106	102	1
	106	106	1

trades	investor_id:int	trade_id:int	rating:int

filteredtradesforuser3	investor_id:int	trade_id:int	rating:int
	3	104	2
	3	101	3

pre_product_matrix	cooccurrence::trade_id:int	cooccurrence::trade_id_2:int	cooccurrence::tradecount:long	filteredtradesforuser3::investor_id:int	filteredtradesforuser3::trade_id:int	filteredtradesforuser3::rating:int
	101	101	2	3	101	3
	102	101	1	3	101	3
	104	101	1	3	101	3
	106	101	1	3	101	3
	101	104	1	3	104	2
	104	104	1	3	104	2

product matrix	trade id:int	trade id 2:int	user3ratingproduct:int
	101	101	6
	102	101	3
	104	101	3
	106	101	3
	101	104	2
	104	104	2

grouped_product_matrix	group:int	product_matrix:bag{:tuple(trade_id:int,trade_id_2:int,user3ratingproduct:int)}
	101	{(101, 101, 6), (101, 104, 2)}
	102	{(102, 101, 3)}
	104	{(104, 101, 3), (104, 104, 2)}
	106	{(106, 101, 3)}

result_Vector	trade_id:int	user3ratingtotal:long
	101	8
	102	3
	104	5
	106	3

filteredtradesforuser3	investor_id:int	trade_id:int	rating:int

joinedrecommendations	result Vector::trade id:int	result Vector::user3ratingtotal:long	filteredtradesforuser3::investor id:int	filteredtradesforuser3::trade id:int	filteredtradesforuser3::rating:int
	101	8	3	101	3
	102	3			
	104	5	3	104	2
	106	3			

filteredrecommendations	result Vector::trade id:int	result Vector::user3ratingtotal:long	filteredtradesforuser3::investor id:int	filteredtradesforuser3::trade id:int	filteredtradesforuser3::rating:int
	102	3			
	106	3			

user3recommendation	trade_id:int	recommendation:long
	102	3
	106	3

user3recommendationsorted	trade_id:int	recommendation:long
	102	3
	106	3

Part3.1 ECL LOADING

Before (Trades.txt on notepad)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

After (trades table on hive)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

Code (trade.ecl)

```
Layout_TradeData := RECORD
INTEGER4 investor_id;
INTEGER4 trade_id;
INTEGER4 rating;
END;

EXPORT trade := DATASET('~online::ik::project::trades',
            Layout_TradeData,
CSV(SEPARATOR('\t'),QUOTE('')));
```

Trades File upload to landing zone and spray

myesp - Enterprise Service

192.168.70.129:8010

Apps | Beginner's Guide to ...

HPCC Systems | EclWatch | Enterprise Services Platform

community_4.2.4-3

Clusters
Activity
Scheduler
ECL
Search Workunits
Browse Workunits
ECL Playground
Queries
Browse
Package Maps
Topology
Target Clusters
Cluster Processes
System Servers
DFU Workunits
Search
Browse
DFU Files
Upload/download File
View Data File
Search File Relationships
Browse Space Usage
Search Logical Files
Browse Logical Files
Browse Files by Scope
Spray Fixed
Spray Delimited
Spray XML
Remote Copy
XRef
Tech Preview
ECL Watch
Resources
Browse
My Account

ONLINEnamephonesupd2	10769679	2014-06-08 15:40:42
ONLINEnamephonesupd3	9136971	2014-06-08 15:40:45
ONLINEnamephonesupd4	8177769	2014-06-08 15:40:50
ONLINEnamephonesupd5	7517097	2014-06-08 15:40:54
OnlineLessonPersons	145490312	2014-06-03 23:15:25
OnlinePeople	2295508	2014-06-05 01:00:19
OnlineProperty	25680732	2014-06-05 01:00:23
OnlineTaxdata	47155688	2014-06-05 01:00:27
OnlineVehicle	29705137	2014-06-05 01:00:33
Persons_SlimMaster	47118400	2014-06-21 14:29:40
Persons_SlimUpdate	47118400	2014-06-21 14:29:51
RoxieAccounts	64213534	2014-06-21 14:29:59
RoxiePersons	13042010	2014-06-21 14:30:10
complextimezones.xml	42665	2014-06-08 18:02:15
embeddedxmltimezones	44660	2014-06-08 19:27:33
imdb_movies	14576732	2014-06-08 19:36:43
onlinelookupcsz	600387	2014-06-15 07:58:08
onlinenamephones	55946526	2014-06-15 07:59:10
onlinenamephonesupd	49535334	2014-06-15 07:59:25
onlinenestedchildxml	89007700	2014-06-08 18:12:35
onlinepersons_slim	94236800	2014-06-15 07:59:57
timezones.xml	52433	2014-06-08 17:43:45
trades.txt	205	2014-07-29 19:01:05

Delete

myesp - Enterprise Service

192.168.70.129:8010

Apps | Beginner's Guide to ...

HPCC Systems | EclWatch | Enterprise Services Platform

community_4.2.4-3

Browse Workunits
ECL Playground
Queries
Browse
Package Maps
Topology
Target Clusters
Cluster Processes
System Servers
DFU Workunits
Search
Browse
DFU Files
Upload/download File
View Data File
Search File Relationships
Browse Space Usage
Search Logical Files
Browse Logical Files
Browse Files by Scope
Spray Fixed
Spray Delimited
Spray XML
Remote Copy
XRef
Tech Preview
ECL Watch
Resources
Browse
My Account
My Account
Change Password
Users/Permissions
Users
Groups
Permissions

online::ik::key::lookup_csz_pay	458,752	20,703	2014-06-29 18:35:43	hp
online::ik::key::lookupcsz	253,952	20,703	2014-06-15 08:45:59	hp
online::ik::key::lookupcsz	253,952	20,703	2014-06-15 08:45:59	hp
online::ik::out::peopleall	105,950,795	27,994	2014-06-06 19:23:42	hp
online::ik::out::weeklyrollup1	19,906,650	214,050	2014-06-08 16:52:21	hp
online::ik::persist::peoplevehicles__p3369532951	32,140,615	27,994	2014-06-05 03:08:55	hp
online::ik::persist::proptax__p977418904	73,670,210	166,758	2014-06-05 05:12:45	hp
online::ik::project::trades	205		2014-07-29 19:02:34	
online::ik::roxie::lookupcsz	600,387	20,703	2014-06-15 08:05:47	
online::ik::roxie::lookupcsz	600,387	20,703	2014-06-15 08:05:47	
online::ik::roxie::namephones	55,946,526	247,551	2014-06-15 08:04:55	
online::ik::roxie::namephonesupd	49,535,334	532,638	2014-06-15 08:06:35	
online::ik::roxie::personsslim	94,236,800	841,400	2014-06-15 08:02:49	
online::ik::roxie::personsslim	94,236,800	841,400	2014-06-15 08:02:49	
online::ik::sf::newbaserollup1	49,619,499	533,543	2014-06-08 17:09:42	hp

Trades.ecl code and output

```
Layout_TradeData := RECORD
INTEGER4 investor_id;
INTEGER4 trade_id;
INTEGER4 rating;
END;

EXPORT trade := DATASET('~online::ik::project::trades',
      Layout_TradeData, CSV(SEPARATOR('\t'),QUOTE('')));
```

##	investor_id	trade_id	rating
1	1	101	1
2	2	101	2
3	3	101	3
4	4	101	4
5	5	101	5
6	6	101	5
7	2	102	1
8	3	102	2
9	5	102	3
10	1	103	1
11	2	103	2
12	3	103	3
13	5	103	4
14	1	104	1
15	3	104	2
16	5	104	4
17	6	104	5
18	4	105	1
19	5	105	2
20	6	105	5
21	1	106	1
22	5	106	2

Part3.2 ECL CO-OCCURRENCE

Before (trades table)

investor_id	trade_id	rating
1	101	1
2	101	2
3	101	3
4	101	4
5	101	5
6	101	5
2	102	1
3	102	2
5	102	3
1	103	1
2	103	2
3	103	3
5	103	4
1	104	1
3	104	2
5	104	4
6	104	5
4	105	1
5	105	2
6	105	5
1	106	1
5	106	2
6	107	1

After (cooccurrence table)

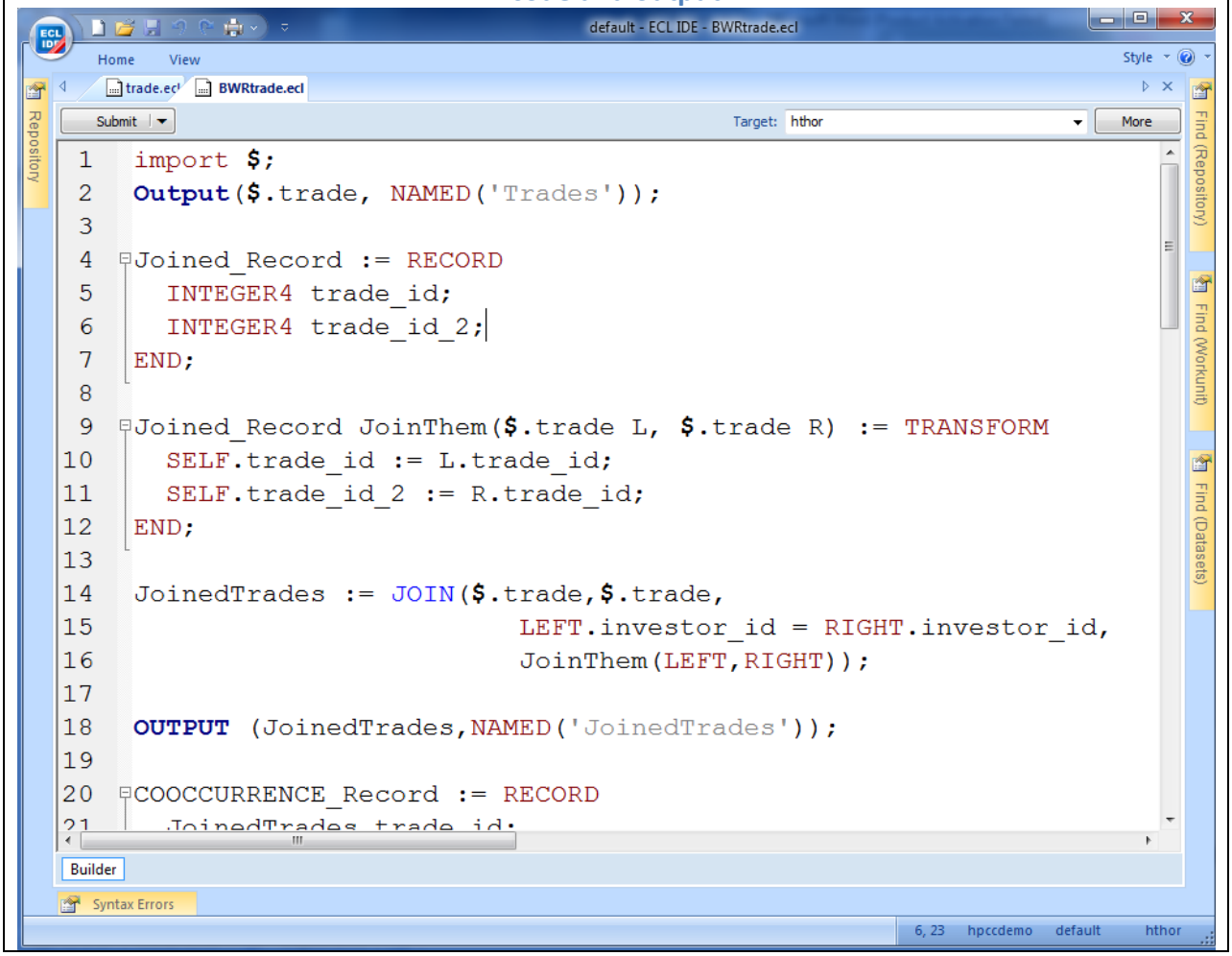
	101	102	103	104	105	106	107
101	6	3	4	4	3	2	1
102	3	3	3	2	1	1	
103	4	3	4	3	1	2	
104	4	2	3	4	2	2	1
105	3	1	1	2	3	1	1
106	2	1	2	2	1	2	
107	1			1	1		1

Code

```
import $;
Output($.trade, NAMED('Trades'));

Joined_Record := RECORD
    INTEGER4 trade_id;
    INTEGER4 trade_id_2;
END;

Joined_Record JoinThem($.trade L, $.trade R) := TRANSFORM
    SELF.trade_id := L.trade_id;
    SELF.trade_id_2 := R.trade_id;
END;

JoinedTrades := JOIN($.trade,$.trade,
                                LEFT.investor_id = RIGHT.investor_id,
                                          JoinThem(LEFT,RIGHT));

OUTPUT (JoinedTrades,NAMED('JoinedTrades'));
```

```
COOCCURRENCE_Record := RECORD
    JoinedTrades.trade_id;
    JoinedTrades.trade_id_2;
    tradescount :=COUNT(GROUP);
END;


COOCCURRENCE_Matrix:=TABLE(JoinedTrades,COOCCURRENCE_Record,JoinedTrades.trad
e_id,JoinedTrades.trade_id_2);
output (COOCCURRENCE_Matrix,NAMED('COOCCURRENCE_Matrix'));
```

code and output



default - ECL IDE - BWRtrade.ecl

Results | Home | View | Format | Style

trade.ecl | BWRtrade.ecl

##	trade_id	trade_id_2
1	101	101
2	101	103
3	101	104
4	101	106
5	103	101
6	103	103
7	103	104
8	103	106
9	104	101
10	104	103
11	104	104
12	104	106
13	106	101
14	106	103
15	106	104
16	106	106
17	101	101
18	101	102
19	101	103
20	102	101
21	102	102
22	102	103
23	103	101
24	103	102
25	103	103

Graphs | Trades | JoinedTrades | COOCCURRENCE_Matrix | filteredtradesforuser3 | product_matrix | recommendations | filtered_result | sorted_filtered_result

Builder | BWRtrade (W20140731-005655)

Syntax Errors

6, 23 hpccdemo default hthor

default - ECL IDE - BWRtrade.ecl

Results | Home | View | Format | Style

trade.ecl | BWRtrade.ecl

##	trade_id	trade_id_2	tradescount
1	101	101	6
2	101	102	3
3	101	103	4
4	101	104	4
5	101	105	3
6	101	106	2
7	101	107	1
8	102	101	3
9	102	102	3
10	102	103	3
11	102	104	2
12	102	105	1
13	102	106	1
14	103	101	4
15	103	102	3
16	103	103	4
17	103	104	3
18	103	105	1
19	103	106	2
20	104	101	4
21	104	102	2
22	104	103	3
23	104	104	4
24	104	105	2
25	104	106	2

Graphs | Trades | JoinedTrades | COOCCURRENCE_Matrix | filteredtradesforuser3 | product_matrix | recommendations | filtered_result | sorted_filtered_result

Builder | BWRtrade (W20140731-005655)

Syntax Errors

6, 23 hpccdemo default hthor

Part3.3 ECL MULTIPLICATION Part 1

Before (cooccurrence table)

	101	102	103	104	105	106	107
101	6	3	4	4	3	2	1
102	3	3	3	2	1	1	
103	4	3	4	3	1	2	
104	4	2	3	4	2	2	1
105	3	1	1	2	3	1	1
106	2	1	2	2	1	2	
107	1			1	1		1

Before(user 3 vector)

101	3
102	2
103	3
104	2
105	0
106	0
107	0

After (product_matrix table)

	101	102	103	104	105	106	107
101	18	6	12	8	0	0	0
102	9	6	9	4	0	0	0
103	12	6	12	6	0	0	0
104	12	4	9	8	0	0	0
105	9	2	3	4	0	0	0
106	6	2	6	4	0	0	0
107	3	0	0	2	0	0	0

Code

```
filteredtradesforuser3 := $.trade(investor_id = 3);
output (filteredtradesforuser3,NAMED('filteredtradesforuser3'));

product_Record := RECORD
    COOCCURRENCE_Record.trade_id;
    COOCCURRENCE_Record.trade_id_2;
    INTEGER4 user3ratingproduct;
END;

product_Record MutiplyThem(COOCCURRENCE_Matrix L, filteredtradesforuser3 R) :=
TRANSFORM
    SELF.trade_id:= L.trade_id;
    SELF.trade_id_2:= L.trade_id_2;
    SELF.user3ratingproduct := L.tradescount * R.rating;
END;

product_matrix   := JOIN(COOCCURRENCE_Matrix,filteredtradesforuser3,
                                 LEFT.trade_id_2 = RIGHT.trade_id,
                                                 MutiplyThem(LEFT,RIGHT));
OUTPUT (product_matrix,NAMED('product_matrix'));
```

code and output

```
30  filteredtradesforuser3 := $.trade(investor_id = 3);
31  output (filteredtradesforuser3,NAMED('filteredtradesforuser3'));
32
33  product_Record := RECORD
34    COOCCURRENCE_Record.trade_id;
35    COOCCURRENCE_Record.trade_id_2;
36    INTEGER4 user3ratingproduct;
37  END;
38
39  product_Record MutiplyThem(COOCCURRENCE_Matrix L, filteredtrades
40    SELF.trade_id:= L.trade_id;
41    SELF.trade_id_2:= L.trade_id_2;
42    SELF.user3ratingproduct := L.tradescount * R.rating;
43  END;
44
45  product_matrix    := JOIN(COOCCURRENCE_Matrix,filteredtradesforus
46                            LEFT.trade_id_2 = RIGHT.trade_id,
47                            MutiplyThem(LEFT,RIGHT));
48  OUTPUT (product_matrix,NAMED('product_matrix'));
49
50  result_Vector := RECORD
```

default - ECL IDE - BWRtrade.ecl

Results | Home | View | Format | Style

trade.ecl | BWRtrade.ecl

##	investor_id	trade_id	rating
1	3	101	3
2	3	102	2
3	3	103	3
4	3	104	2

Graphs | Trades | JoinedTrades | COOCCURRENCE_Matrix | filteredtradesforuser3 | product_matrix | recommendations | filtered_result | sorted_filtered_result

Builder | BWRtrade (W20140731-005655)

Syntax Errors

hpccdemo | default | hthor

default - ECL IDE - BWRtrade.ecl

Results | Home | View | Format | Style

trade.ecl | BWRtrade.ecl

##	trade_id	trade_id_2	user3ratingproduct
1	101	101	18
2	102	101	9
3	103	101	12
4	104	101	12
5	105	101	9
6	106	101	6
7	107	101	3
8	101	102	6
9	102	102	6
10	103	102	6
11	104	102	4
12	105	102	2
13	106	102	2
14	101	103	12
15	102	103	9
16	103	103	12
17	104	103	9
18	105	103	3
19	106	103	6
20	101	104	8
21	102	104	4
22	103	104	6
23	104	104	8
24	105	104	4
25	106	104	4

Graphs | Trades | JoinedTrades | COOCCURRENCE_Matrix | filteredtradesforuser3 | product_matrix | recommendations | filtered_result | sorted_filtered_result

Builder | BWRtrade (W20140731-005655)

Syntax Errors

hpccdemo | default | hthor

Part3.3 ECL MULTIPLICATION Part 2

Before (product_matrix table)

	101	102	103	104	105	106	107
101	18	6	12	8	0	0	0
102	9	6	9	4	0	0	0
103	12	6	12	6	0	0	0
104	12	4	9	8	0	0	0
105	9	2	3	4	0	0	0
106	6	2	6	4	0	0	0
107	3	0	0	2	0	0	0

After (result_Vector table)

101	44
102	28
103	36
104	33
105	18
106	18
107	5

Code

```
result_Vector := RECORD
    product_matrix.trade_id;
    INTEGER4 user3ratingtotal := SUM(GROUP,product_matrix.user3ratingproduct);
END;


recommendations := TABLE(product_matrix,result_Vector,trade_id);
OUTPUT (recommendations,NAMED('recommendations'));
```

code and output

```
result_Vector := RECORD
  product_matrix.trade_id;
  INTEGER4 user3ratingtotal := SUM(GROUP,product_matrix.user3rat
END;


recommendations := TABLE(product_matrix,result_Vector,trade_id);
OUTPUT (recommendations,NAMED('recommendations'));

filtered_result_record := RECORD
  INTEGER4 trade_id;
  INTEGER4 user3ratingtotal;
END;

filtered_result_record JoinRecs(result_Vector L, filteredtradesf
  SELF.trade_id := L.trade_id;
  SELF.user3ratingtotal := L.user3ratingtotal;
END;

filtered_result := JOIN(recommendations,filteredtradesforuser3,
                        LEFT.trade_id = RIGHT.trade_id
```

##	trade_id	user3ratingtotal
1	101	44
2	102	28
3	103	36
4	104	33
5	105	18
6	106	18
7	107	5

Part3.4 ECL RECOMMENDATIONS

Before (result_Vector table)

101	44
102	28
103	36
104	33
105	18
106	18
107	5

After (User3_recommendations table)

105	18
106	18
107	5

Code

```
filtered_result_record := RECORD
    INTEGER4 trade_id;
    INTEGER4 user3ratingtotal;
END;

filtered_result_record JoinRecs(result_Vector L, filteredtradesforuser3 R) :=
TRANSFORM
    SELF.trade_id := L.trade_id;
    SELF.user3ratingtotal := L.user3ratingtotal;
END;

filtered_result := JOIN(recommendations,filteredtradesforuser3,
                                LEFT.trade_id = RIGHT.trade_id,
                                            JoinRecs(LEFT,RIGHT),
                                LEFT ONLY);
OUTPUT (filtered_result,NAMED('filtered_result'));

sorted_filtered_result := Sort(filtered_result,-user3ratingtotal);
OUTPUT (sorted_filtered_result,NAMED('sorted_filtered_result'));
```

code and output

```
58
59 filtered_result_record := RECORD
60   INTEGER4 trade_id;
61   INTEGER4 user3ratingtotal;
62 END;
63
64 filtered_result_record JoinRecs(result_Vector L, filteredtradesf
65   SELF.trade_id := L.trade_id;
66   SELF.user3ratingtotal := L.user3ratingtotal;
67 END;
68
69 filtered_result := JOIN(recommendations,filteredtradesforuser3,
70                        LEFT.trade_id = RIGHT.trade_id,
71                        JoinRecs(LEFT,RIGHT),
72                        LEFT ONLY);
73 OUTPUT (filtered_result,NAMED('filtered_result'));
74
75 sorted_filtered_result := Sort(filtered_result,-user3ratingtotal
76 OUTPUT (sorted_filtered_result,NAMED('sorted_filtered_result'));
77
```

default - ECL IDE - BWRtrade.ecl

Results | Home | View | Format | Style

trade.ecl | BWRtrade.ecl

##	trade_id	user3ratingtotal
1	105	18
2	106	18
3	107	5

Graphs | Trades | JoinedTrades | COOCCURRENCE_Matrix | filteredtradesforuser3 | product_matrix | recommendations | filtered_result | sorted_filtered_result

Builder | BWRtrade (W20140731-005655)

Syntax Errors

hpccdemo default hthor

default - ECL IDE - BWRtrade.ecl

Results | Home | View | Format | Style

trade.ecl | BWRtrade.ecl

##	trade_id	user3ratingtotal
1	105	18
2	106	18
3	107	5

Graphs | Trades | JoinedTrades | COOCCURRENCE_Matrix | filteredtradesforuser3 | product_matrix | recommendations | filtered_result | sorted_filtered_result

Builder | BWRtrade (W20140731-005655)

Syntax Errors

hpccdemo default hthor

Part4 Pig Implementation Using Amazon AWS

The following screen captures illustrate the steps taken to run pig script of amazon web services. The steps can be summarized as follows:

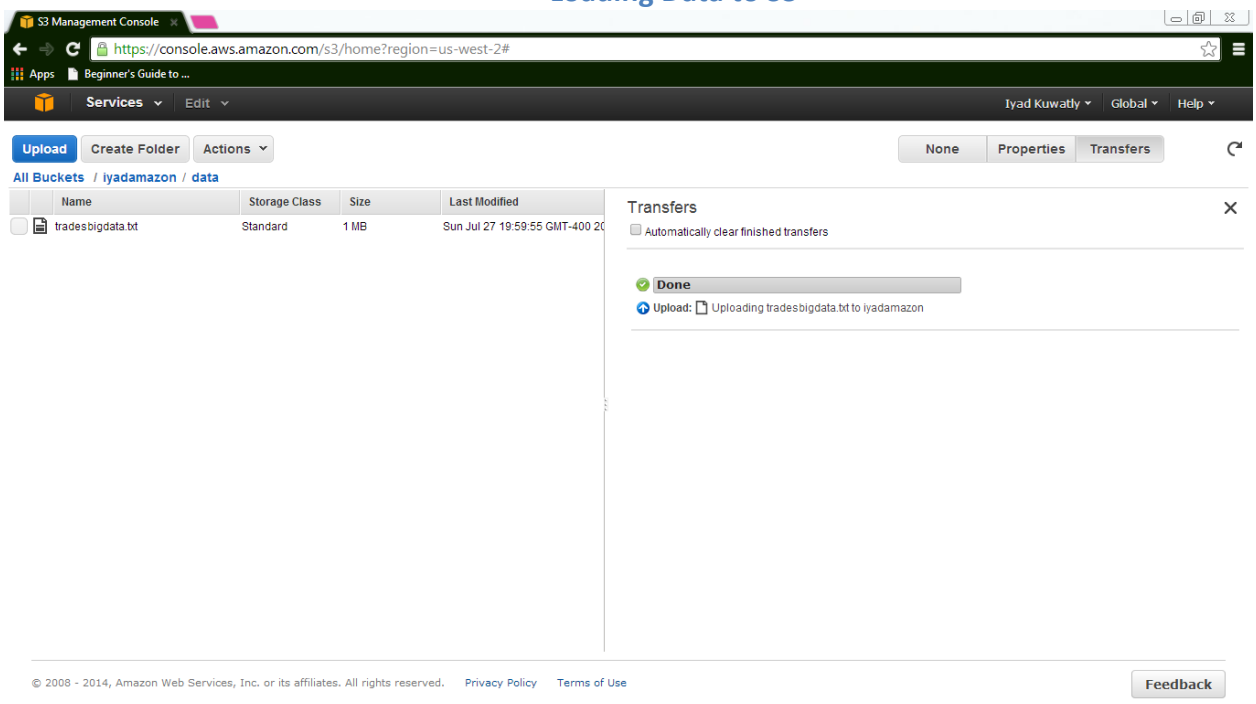
1- Creating AWS account
2- Uploading data and pig script into S3 (Amazon Scalable Storage in the cloud)
3- Configuring the cluster and Creating Elastic MapReduce Job
4- Monitoring cluster starting and job processing
5- Viewing results and logs after job completion
6- Checking the costs by viewing the bill

The following is the Pig code used in AWS:

```
trades = LOAD 's3://iyadamazon/data/tradesbigdata.txt' USING PigStorage('\t')
AS (investor_id:int, trade_id:int, rating:int) ;

trades_2 = FOREACH trades GENERATE investor_id AS investor_id_2, trade_id AS
trade_id_2, rating AS rating_2 ;
joinedtradeds = JOIN trades BY investor_id, trades_2 BY investor_id_2 ;
groupedtrades = group  joinedtradeds  by (trade_id,trade_id_2);
cooccurrence = FOREACH groupedtrades GENERATE
        group.trade_id as trade_id,
        group.trade_id_2 as trade_id_2,
        COUNT($1) as tradecount;

filteredtradesforuser3 = filter trades BY investor_id == 3;
pre_product_matrix = JOIN cooccurrence BY trade_id_2, filteredtradesforuser3
BY trade_id ;
product_matrix = FOREACH pre_product_matrix GENERATE
        $0 as trade_id,
        $1 as trade_id_2,
        (int)$2*$5 as user3ratingproduct;
grouped_product_matrix = group  product_matrix  by trade_id;


result_Vector = FOREACH grouped_product_matrix GENERATE
        $0 as trade_id,
        SUM(product_matrix.user3ratingproduct) as user3ratingtotal;

joinedrecommendations = JOIN result_Vector by trade_id LEFT,
filteredtradesforuser3 BY trade_id;
```

```
filteredrecommendations = filter joinedrecommendations BY $2 is null;

user3recommendation = FOREACH filteredrecommendations GENERATE
$0 as trade_id,
$1 as recommendation;

user3recommendationsorted = order user3recommendation by recommendation desc;

store user3recommendationsorted into
's3://iyadamazon/results/tradesrecommendations.txt' using PigStorage('\t');
```

Loading Data to S3



Loading Pig Script to S3

S3 Management Console | excel - Export from pig to

https://console.aws.amazon.com/s3/home?region=us-west-2

Apps | Beginner's Guide to ...

Services | Edit | Iyad Kuwatly | Global | Help

Upload | Create Folder | Actions | None | Properties | Transfers

All Buckets / iyadamazon / code

Name	Storage Class	Size	Last Modified
pigcodeaws.pig	Standard	1.4 KB	Sun Jul 27 20:12:55 GMT-400 20

Transfers

Automatically clear finished transfers

Done
Upload: Uploading pigcodeaws.pig to iyadamazon

Done
Delete: Deleting pigcodeaws.pig from results

Done
Upload: Uploading pigcodeaws.pig to iyadamazon

 Privacy Policy | Terms of Use | Feedback

Configuring the cluster

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#create-cluster:

Apps | Beginner's Guide to ...

Services | Edit | Iyad Kuwatly | Oregon | Help

Elastic MapReduce | Create Cluster | EMR Help

Configure sample application

Cluster Configuration

Cluster name: bigdataiyadcluster

Termination protection: Yes / No — Prevents accidental termination of the cluster: to shut down the cluster, you must turn off termination protection. Learn more

Logging: Enabled — Copy the cluster's log files automatically to S3. Learn more

Log folder S3 location

s3://iyadamazon/logs/

s3://<bucket-name>/<folder>/

Debugging: Enabled — Index logs to enable console debugging functionality (requires logging). Learn more

Tags

Optional: Add up to 10 tags to your EMR cluster. A tag consists of a case-sensitive key-value pair. Tags on EMR clusters are propagated to the underlying EC2 instances. Learn more about tagging your Amazon EMR clusters.

Key	Value (optional)
Add a key to create a tag	

Software Configuration

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#create-cluster:

Apps | Beginner's Guide to ...

Software Configuration

Hadoop distribution Amazon — Use Amazon's Hadoop distribution. Learn more

AMI version 3.1.0 — Determines the base configuration of the instances in your cluster, including the Hadoop version. Learn more

MapR — Use MapR's Hadoop distribution. Learn more

Applications to be installed	Version
Hive	0.11.0.2
Pig	0.12.0

Additional applications Select an application

Configure and add

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 instances, complete this form. Request Spot instances (unused EC2 capacity) to save money.

Network vpc-8fe71bea (172.31.0.0/16) (default) — Use a Virtual Private Cloud (VPC) to process sensitive data or connect to a private network. Create a VPC

EC2 Subnet No preference (random subnet) — Create a Subnet

	EC2 instance type	Count	Request spot	
Master	m1.medium	1		The Master instance assigns Hadoop tasks to core and task nodes, and monitors their status.
Core	m1.medium	2		Core instances run Hadoop tasks and store data using the Hadoop Distributed File System (HDFS).
Task	m1.medium	0		Task instances run Hadoop tasks.

Security and Access

EC2 key pair iyadkey — Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". Learn more

IAM user access All other IAM users / No other IAM users — Control the visibility of this cluster to other IAM users. Learn more

IAM Roles

An IAM role for the EMR service and an EC2 instance profile for instances in an EMR cluster are recommended. You can create and assign these roles to limit the permissions of the EMR service and applications running on a cluster.

EMR role EMR_DefaultRole — Allows EMR to access other AWS Services such as EC2 on your behalf. Learn more

EC2 instance profile EMR_EC2_DefaultRole — Allows EC2 instances in an EMR cluster to access other AWS services such as S3. Learn more

Bootstrap Actions

Bootstrap actions are scripts that are executed during setup before Hadoop starts on every cluster node. You can use them to install additional software and customize your applications. Learn more

Bootstrap action type	Name	S3 location	Optional arguments

Add bootstrap action Configure Hadoop

Configure and add

Steps

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#create-cluster:

Apps Beginner's Guide to ...

Add bootstrap action: Configure Hadoop

Configure and add

Steps

A step is a unit of work you submit to the cluster. A step might contain one or more Hadoop jobs, or contain instructions to install or configure an application. You can submit up to 256 steps to a cluster. Learn more

Name	Action on failure	JAR S3 location	Arguments
BigDataCoursePigProgram	Continue	s3://us-west-2.elasticmapreduce/libs/script-runner/script-runner.jar	s3://us-west-2.elasticmapreduce/libs/pig/pig-script --run-pig-script --pig-versions 0.12.0 --args -f s3://iyadamazon/code/pigcodeaws.pig -p INPUT=s3://iyadamazon/data/tradesbigdata.txt -p OUTPUT=s3://iyadamazon/results/

Add step: Pig program

Configure and add

Auto-terminate: Yes — Automatically terminate cluster after the last step is completed.

No — Keep cluster running until you terminate it.

Cancel Create cluster

Feedback

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#create-cluster:

Apps Beginner's Guide to ...

EC2 instance profile: EMR_EC2_DefaultRole — AWS services such as S3. Learn more

Bootstrap Actions

Add Step

Step type: Pig program

Name: BigDataCoursePigProgram

Script S3 location*: s3://iyadamazon/code/pigcodeaws.pig — S3 location of your Pig script.
s3://<bucket-name>/<path-to-file>

Input S3 location: s3://iyadamazon/data/tradesbigdata.txt — S3 location of your Pig input files.
s3://<bucket-name>/<folder>/

Output S3 location: s3://iyadamazon/results/ — S3 location of your Pig output files.
s3://<bucket-name>/<folder>/

Arguments — Specify optional arguments for your script.

Action on failure: Continue — What to do if the step fails.

Cancel Add

Cancel Create cluster

Feedback

Monitoring Progress

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-details:j-YXAGOJWS300M

Apps | Beginner's Guide to ...

Services | Edit | Iyad Kuwatly | Oregon | Help

Elastic MapReduce > Cluster List > Cluster Details | EMR Help

Add step | Resize | Clone | Terminate

Cluster: bigdataiyadcluster Starting

Master public DNS: --

Tags: -- View All / Edit

Summary

ID: j-YXAGOJWS300M
Creation date: 2014-07-27 20:15 (UTC-4)
Elapsed time:
Auto-terminate: Yes
Termination protection: On Change

Configuration Details

AMI version: 3.1.0
Hadoop distribution: Amazon 2.4.0
Applications: Hive 0.11.0.2, Pig 0.12.0
Log URI: s3://iyadamazon/logs/

Security/Network

Availability zone: --
Subnet ID: subnet-f07e71b6
Key name: iyadkey
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: None Change

Hardware

Master: Provisioning 1 m1.medium
Core: Provisioning 2 m1.medium
Task: --

Monitoring

Steps

Bootstrap Actions

Feedback

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-details:j-YXAGOJWS300M

Apps | Beginner's Guide to ...

Auto-terminate: Yes
Termination protection: On Change

Applications: Hive 0.11.0.2, Pig 0.12.0
Log URI: s3://iyadamazon/logs/

Key name: iyadkey
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: None Change

Monitoring

Steps

Add step

Steps

View all interactive jobs | View all jobs

Filter: All steps | Filter steps ... | 4 steps (all loaded)

ID	Name	Status	Start time (UTC-4)	Elapsed time	Log files	Actions
s-33UXB0SLSBG3T	Setup hive	Pending			View logs	View jobs
s-1Z6Y4W4N18M2K	Setup hadoop debugging	Pending			View logs	View jobs
s-3TL146NP63L5A	Setup pig	Pending			View logs	View jobs
s-10N04S9B4XRF7	BigDataCourse PigProgram	Pending			View logs	View jobs

Bootstrap Actions

Feedback

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-details:j-YXAGOJWS300M

Apps | Beginner's Guide to ...

Services | Edit | Iyad Kuwatly | Oregon | Help

Elastic MapReduce > Cluster List > Cluster Details | EMR Help

Add step | Resize | Clone | Terminate

Cluster: bigdataiyadcluster Running Running step

Master public DNS: ec2-54-191-175-103.us-west-2.compute.amazonaws.com

Tags: -- View All / Edit

Summary

ID: j-YXAGOJWS300M
Creation date: 2014-07-27 20:15 (UTC-4)
Elapsed time: 4 minutes
Auto-terminate: Yes
Termination protection: On Change

Configuration Details

AMI version: 3.1.0
Hadoop distribution: Amazon 2.4.0
Applications: Hive 0.11.0.2, Pig 0.12.0
Log URI: s3://iyadamazon/logs/

Security/Network

Availability zone: us-west-2c
Subnet ID: subnet-f07e71b6
Key name: iyadkey
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: None Change

Hardware

Master: Running 1 m1.medium
Core: Running 2 m1.medium
Task: --

Monitoring

Steps

Bootstrap Actions

Name	Location	Optional arguments

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-details:j-YXAGOJWS300M

Apps | Beginner's Guide to ...

Monitoring

Steps

Add step

Steps

View all interactive jobs | View all jobs

Filter: All steps | Filter steps ... | 4 steps (all loaded)

ID	Name	Status	Start time (UTC-4)	Elapsed time	Log files	Actions
s-10N04S9B4XRF7	BigDataCourse PigProgram	Running	2014-07-27 20:21	1 minute	View logs	View jobs
s-3TL146NP63L5A	Setup pig	Completed	2014-07-27 20:21	17 seconds	View logs	View jobs
s-33UXB0SLSBG3T	Setup hive	Completed	2014-07-27 20:21	23 seconds	View logs	View jobs
s-1Z6Y4W4N18M2K	Setup hadoop debugging	Completed	2014-07-27 20:20	31 seconds	View logs	View jobs

JAR location: s3://us-west-2.elasticmapreduce/libs/script-runner/script-runner.jar
Main class: None
Arguments: s3://us-west-2.elasticmapreduce/libs/pig/pig-script --run-pig-script --pig-versions 0.12.0 --args -f s3://iyadamazon/code/pigcodeaws.pig -p INPUT=s3://iyadamazon/data/tradesbigdata.txt -p OUTPUT=s3://iyadamazon/results/
Action on failure: Continue

Bootstrap Actions

Feedback

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-list:

Apps | Beginner's Guide to ...

Services | Edit | Iyad Kuwatly | Oregon | Help

Elastic MapReduce | Cluster List | EMR Help

Create cluster | View details | Clone | Terminate

Filter: All clusters | Filter clusters ... | 3 clusters (all loaded)

Name	ID	Status	Creation time (UTC-4)	Elapsed time	Normalized instance hours
bigdataiyadcluster	j-YXAGOJWS300M	Running	2014-07-27 20:15	9 minutes	6
pigtest	j-1Q051OAK3G8HH	Terminated User request	2014-07-19 17:55	39 seconds	0
iyadcluster	j-3XBZFTXLV6R0	Terminated User request	2014-07-17 18:41	52 minutes	6

Summary

Master public DNS: ec2-54-191-175-103.us-west-2.compute.amazonaws.com

Termination protection: On Change

Tags: -- View All / Edit

Hardware | Resize

Master: Running 1 m1.medium

Core: Running 2 m1.medium

Task: --

View cluster details

Steps | Add Step | View all interactive jobs

Name	Status	Start time (UTC-4)	Elapsed time
BigDataCoursePigProgram	Running	2014-07-27 20:21	3 minutes
Setup pig	Completed	2014-07-27 20:21	17 seconds
Setup hive	Completed	2014-07-27 20:21	23 seconds
Setup hadoop debugging	Completed	2014-07-27 20:20	31 seconds

Bootstrap Actions

Name

No bootstrap actions available

Feedback

AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-details:j-YXAGOJWS300M

Apps | Beginner's Guide to ...

Creation date: 2014-07-27 20:15 (UTC-4)

Elapsed time: 13 minutes

Auto-terminate: Yes

Termination protection: On Change

Hadoop distribution: Amazon 2.4.0

Applications: Hive 0.11.0.2, Pig 0.12.0

Log URI: s3://iyadamazon/logs/

zone:

Subnet ID: subnet-f07e71b6

Key name: iyadkey

EC2 instance profile: EMR_EC2_DefaultRole

EMR role: EMR_DefaultRole

Visible to all users: None Change

Core: Running 2 m1.medium

Task: --

Monitoring

Steps

Add step

Steps > Jobs > Tasks | View all interactive jobs | View all jobs

Tasks for: s-10N04S9B4XRF7, Job 1406506767534_0003

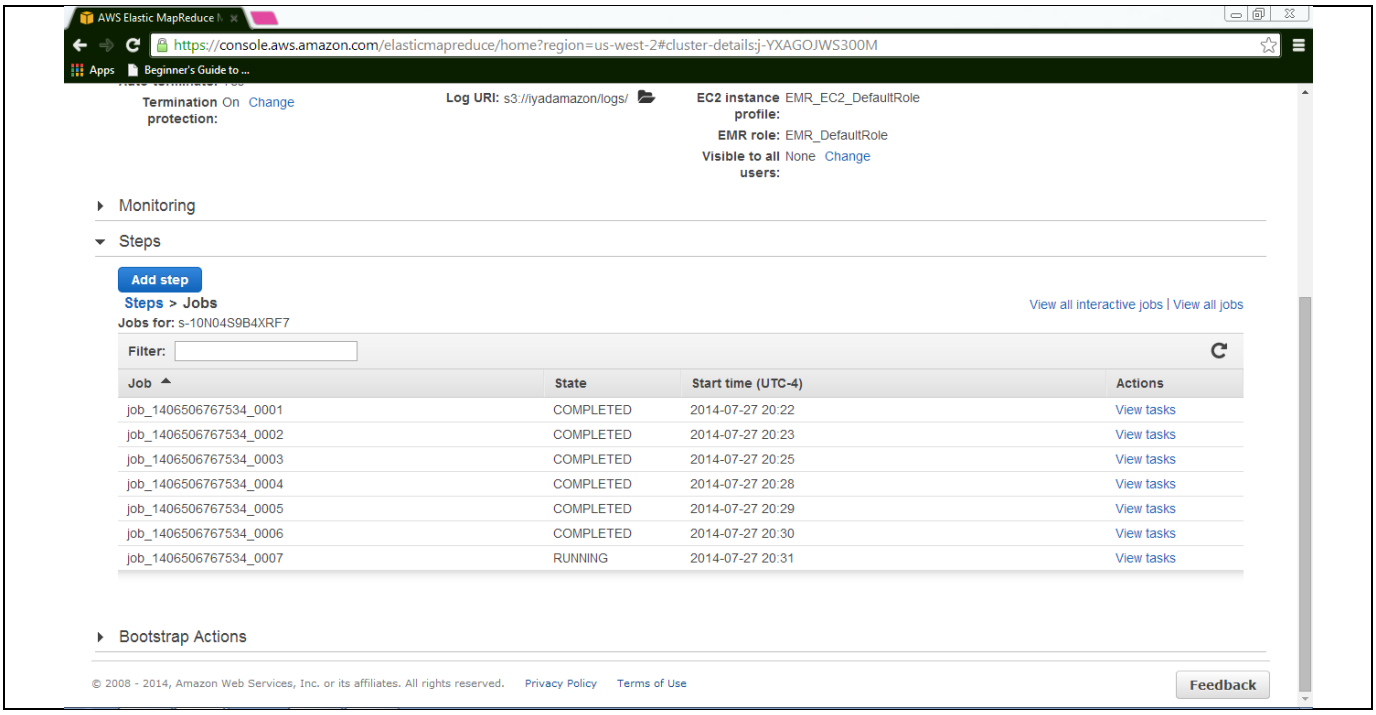
Task summary: 4 total tasks - 4 completed, 0 running, 0 failed, 0 pending, 0 cancelled.

Filter:

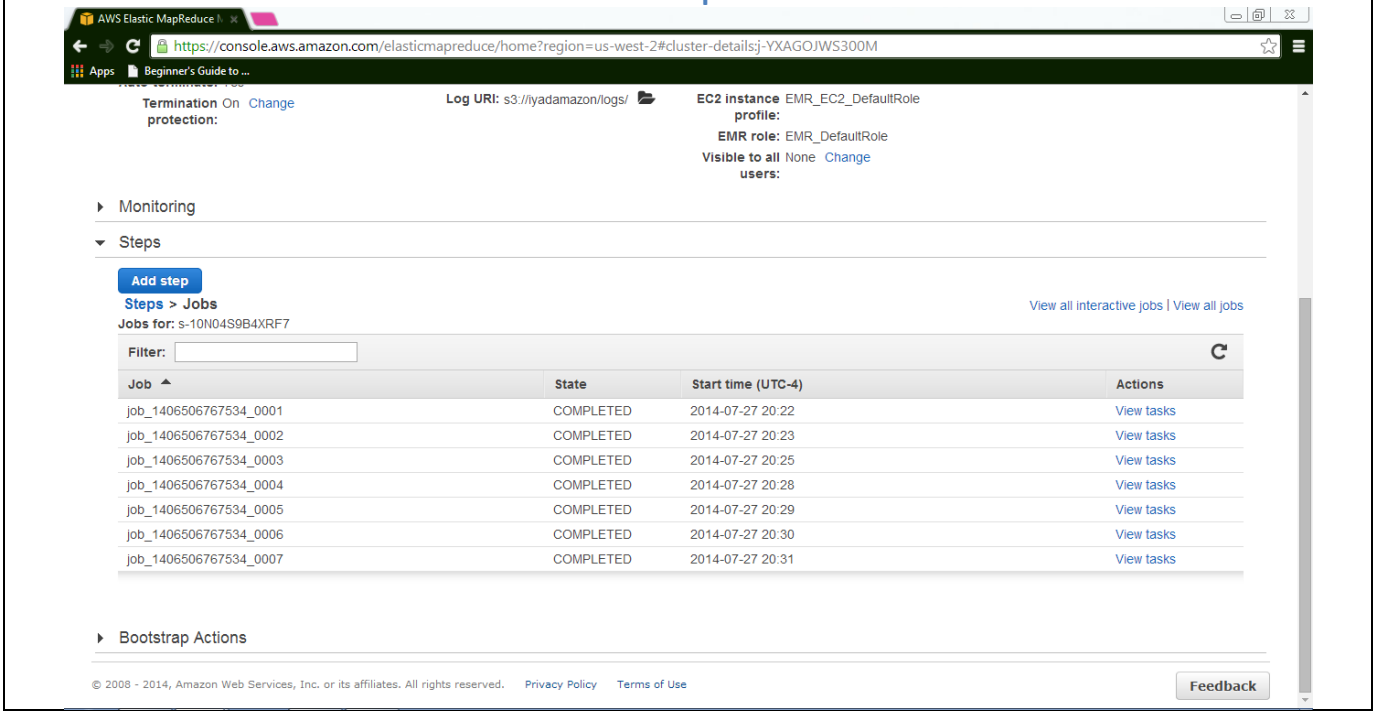
Task	Type	State	Start time (UTC-4)	Actions
r_000000	REDUCE	COMPLETED	2014-07-27 20:27:00	View attempts
m_000002	MAP	COMPLETED	2014-07-27 20:25:29	View attempts
m_000001	MAP	COMPLETED	2014-07-27 20:25:28	View attempts
m_000000	MAP	COMPLETED	2014-07-27 20:25:27	View attempts

Bootstrap Actions

Feedback



Job Completed



AWS Elastic MapReduce

https://console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-details:j-YXAGOJWS300M

Apps | Beginner's Guide to ...

Services | Edit | Iyad Kuwatly | Oregon | Help

Elastic MapReduce | Cluster List > Cluster Details | EMR Help

Add step | Resize | Clone | Terminate

Cluster: bigdataiyadcluster Terminated Steps completed

Master public DNS: ec2-54-191-175-103.us-west-2.compute.amazonaws.com
Tags: --

Summary
- **ID:** j-YXAGOJWS300M
- **Creation date:** 2014-07-27 20:15 (UTC-4)
- **End date:** 2014-07-27 20:36 (UTC-4)
- **Elapsed time:** 20 minutes
- **Auto-terminate:** Yes
- **Termination protection:** On

Configuration Details
- **AMI version:** 3.1.0
- **Hadoop distribution:** Amazon 2.4.0
- **Applications:** Hive 0.11.0.2, Pig 0.12.0
- **Log URI:** s3://iyadamazon/logs/

Security/Network
- **Availability zone:** us-west-2c
- **Subnet ID:** subnet-f07e71b6
- **Key name:** iyadkey
- **EC2 instance profile:** EMR_EC2_DefaultRole
- **EMR role:** EMR_DefaultRole
- **Visible to all users:** None Change

Hardware
- **Master:** Terminated 1 m1.medium
- **Core:** Terminated 2 m1.medium
- **Task:** --

Monitoring

Steps

Bootstrap Actions

Feedback

Examining and downloading results

S3 Management Console

https://console.aws.amazon.com/s3/home?region=us-west-2

Apps | Beginner's Guide to ...

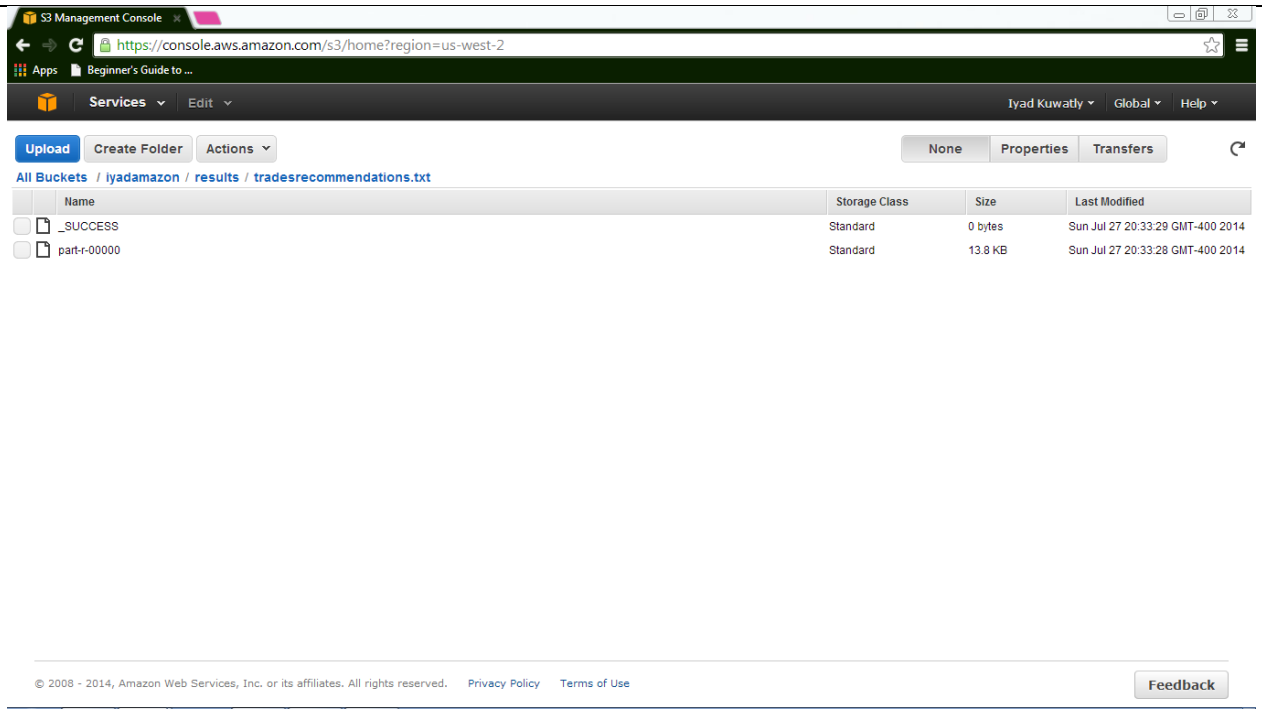
Services | Edit | Iyad Kuwatly | Global | Help

Upload | Create Folder | Actions | None | Properties | Transfers

All Buckets / iyadamazon / results

Name	Storage Class	Size	Last Modified
tradesrecommendations.txt	--	--	--

Feedback



Contents of output file `'s3://iyadamazon/results/tradesrecommendations.txt'`

286	14370
50	14142
100	12511
313	12362
748	10976
269	10746
1	10630
127	10628
121	10573
7	10416
174	10118
117	10054
56	10008
237	9832
98	9557
222	9225
172	9107
405	9022
301	8977
289	8864
79	8840

Examining the charges for the job

Billing Management Cons | Billing Management Cons

https://console.aws.amazon.com/billing/home#/bill?year=2014&month=7

Apps | Beginner's Guide to ...

Services | Edit | Iyad Kuwatly | Global | Help

- Dashboard
- Bills
- Cost Explorer
- Payment Methods
- Payment History
- Consolidated Billing
- Account Settings
- Reports
- Preferences
- Credits
- DevPay

Bills

Date: July 2014

Download CSV | Print

Summary	Amount
AWS Service Charges	$0.34
There are no invoices for the selected month.	

+ Expand All

Details	Total
AWS Service Charges	$0.34
Data Pipeline	$0.01
Data Transfer	$0.00
Elastic Compute Cloud	$0.26
Elastic MapReduce	$0.07
Simple Notification Service	$0.00
Simple Queue Service	$0.00
Simple Storage Service	$0.00
SimpleDB	$0.00
CT to be collected	$0.00

Conclusion

Implementing the recommendation engine in different platforms and in different languages was very useful to have a close feeling of the different big data databases. Specifically it allows understanding how the language and platform are manipulating the data. In hive and ECL is more tables centric where as in Pig it is key, value centric allowing for large space to analyze and restructure the data as needed. Grunt and hive CLIs were very helpful in learning the language going step by step with the code. Batch processing is fast and efficient especially when the script and code has been tested locally on sample of data.