
CS5590 – Big Data (Hadoop/Spark) – Lab Assignment 1 - Report
#10, Avni Mehta

Task 1

Objective: Create a map reduce program to find mutual friends for the following input

A -> B C D
B -> A C D E
C -> A B D E
D -> A B C E
E -> B C D

Algorithm:

I am doing the following steps:

Mapper:

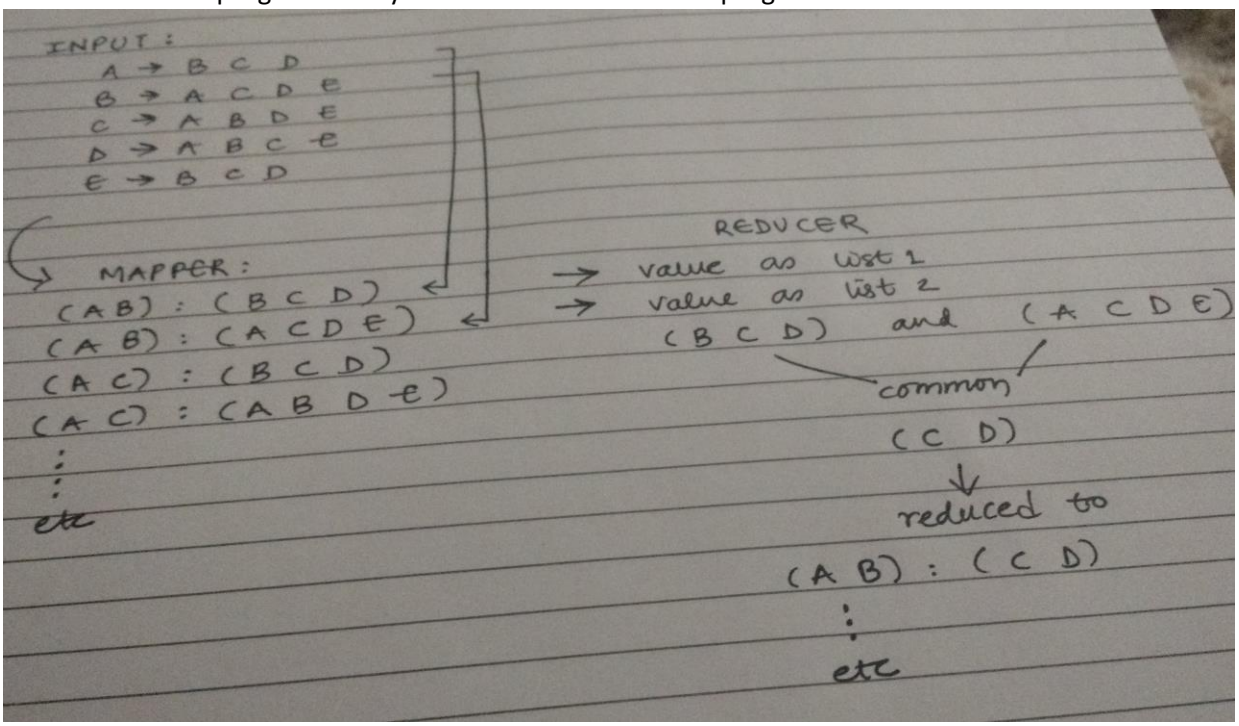
1. Split the input file line by line
2. Split the line by '->' format. The left side of the split gives the user name and the right side gives all the friends.
3. The right split can be further split by space to get a list of friends.
4. For each friend f, store the key as sorted (user, f)
5. The mapper program will output (key, value) where value is the original list of friends from the right split.

Reducer:

1. Store the value into lists.
2. Compare the lists to find common elements
3. Output (key, value) where value is the common friends

Main Program:

The main program is very much like the word count program.



Code:

Comments are provided for each important step in the code.

Mapper:

MutualFriends.java

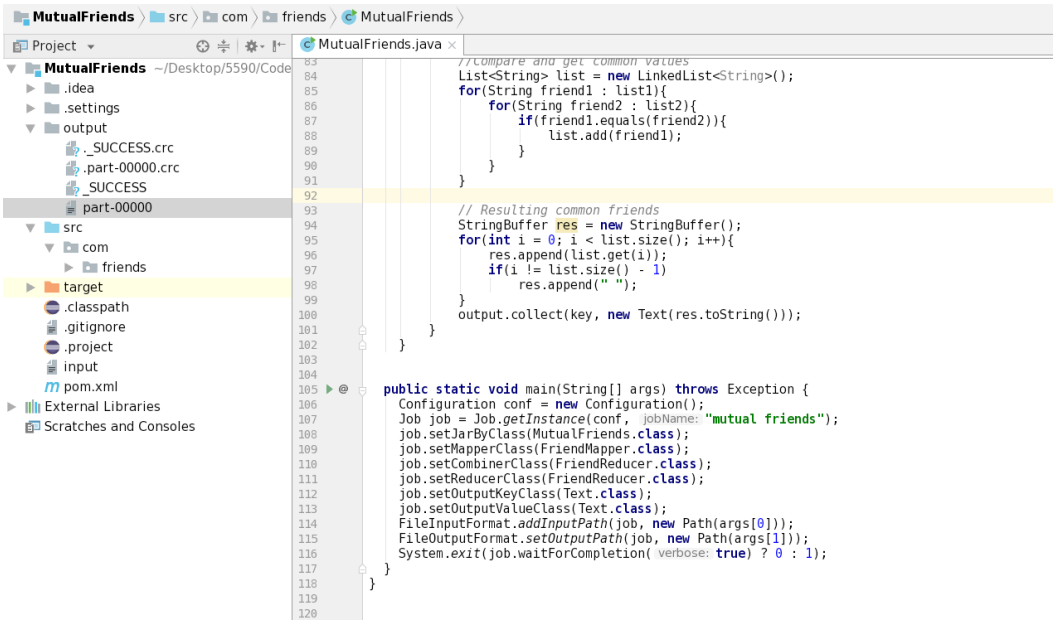
```
    public void map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {

        // Each line separated by newline character
        StringTokenizer itr = new StringTokenizer(value.toString(), delim: "\n");
        String line = null;
        String[] lineSplit = null, friends = null;
        String f1, f2;
        int compare;
        Text keyFriends = new Text();

        while (itr.hasMoreTokens()) {
            // Get the first line
            line = itr.nextToken();

            // Split the line into user and friends by "->"
            // Input:  A -> B C D
            lineSplit = line.split( regex: " -> ");

            // Get friends in array
            friends = lineSplit[1].split( regex: " ");

            // Get the user as f1 (Need to sort)
            f1 = lineSplit[0];

            // Get friends in loop
            for (int i = 0; i < friends.length; i++) {

                // Get friend in variable f2
                f2 = friends[i];

                // Sort
                compare = f1.compareTo(f2);
                if (compare < 0) {
                    keyFriends.set(f1 + " " + f2);
                } else {
                    keyFriends.set(f2 + " " + f1);
                }
                context.write(keyFriends, new Text( string: ": " + lineSplit[1]));
            }
```

Reducer:

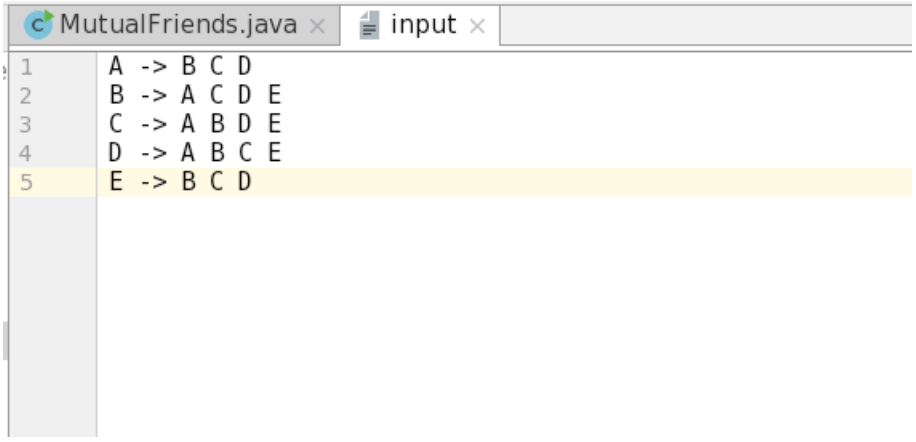
```
public static class FriendReducer extends Reducer<Text, Text, Text, Text>{
    public void reduce(Text key, Iterator<Text> values,
                       OutputCollector<Text, Text> output, Reporter reporter) throws IOException{


        Text[] frndLists = new Text[2];

        int cnt = 0;
        while(values.hasNext()){
            frndLists[cnt++] = new Text(values.next());
        }

        //Get both list values into lists
        String[] list1 = frndLists[0].toString().split( regex: " ");
        String[] list2 = frndLists[1].toString().split( regex: " ");

        //Compare and get common values
        List<String> list = new LinkedList<String>();
        for(String friend1 : list1){
            for(String friend2 : list2){
                if(friend1.equals(friend2)){
                    list.add(friend1);
                }
            }
        }

        // Resulting common friends
        StringBuffer res = new StringBuffer();
        for(int i = 0; i < list.size(); i++){
            res.append(list.get(i));
            if(i != list.size() - 1)
                res.append(" ");
        }
        output.collect(key, new Text(res.toString()));
    }
}
```

Main:

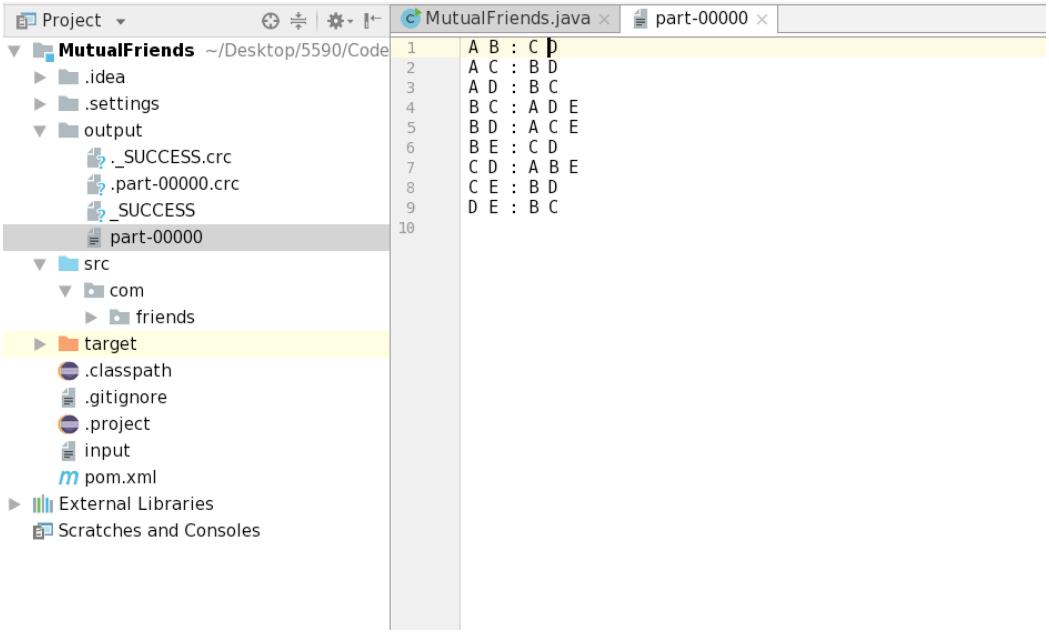
```
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, jobName: "mutual friends");
  job.setJarByClass(MutualFriends.class);
  job.setMapperClass(FriendMapper.class);
  job.setCombinerClass(FriendReducer.class);
  job.setReducerClass(FriendReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(Text.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
}
}
```



Input:



Output:



Task 2

Objective:

a) Consider one of the use case and use a simple dataset. Describe the use case considered based on your assumptions, report the dataset, its fields, datatype etc.

Use Case Considered = Coursera

Problem: Need a high performing and highly scalable database where new features can be easily added.
Solution: They chose Cassandra. But in this task, we will compare both HBase and Cassandra

Dataset:
The table should contain columns like user_id, user_name, course_id, course_name, course_completion_status, course_start_date etc.
The user_id and course_id columns would be numeric and the user_name, course_name and course_completion_status columns would be text. Column course_start_date is date type.

b) Use HBase to implement a Solution for the use case. Report at least 3 queries, their input and output. The query's relevance towards solving the use case is important.

HBase table creation:

```
create 'user_courses', 'user_info', 'course_info'
```

HBase insert rows:

```
put 'user_courses', '1', 'user_info:user_id', '2341'
put 'user_courses', '1', 'user_info:user_name', 'Avni Mehta'

put 'user_courses', '1', 'course_info:course_id', '21'
put 'user_courses', '1', 'course_info:course_name', 'Python'
put 'user_courses', '1', 'course_info:course_completion_status', 'In Progress'
put 'user_courses', '1', 'course_info:course_start_date', '04/23/2018'

put 'user_courses', '2', 'user_info:user_id', '2341'
put 'user_courses', '2', 'user_info:user_name', 'Avni Mehta'

put 'user_courses', '2', 'course_info:course_id', '341'
put 'user_courses', '2', 'course_info:course_name', 'AWS'
put 'user_courses', '2', 'course_info:course_completion_status', 'Complete'
put 'user_courses', '2', 'course_info:course_start_date', '06/18/2018'

put 'user_courses', '3', 'user_info:user_id', '10'
put 'user_courses', '3', 'user_info:user_name', 'Hardik Mehta'

put 'user_courses', '3', 'course_info:course_id', '21'
put 'user_courses', '3', 'course_info:course_name', 'Python'
```

put 'user_courses', '3', 'course_info:course_completion_status', 'Complete'
put 'user_courses', '3', 'course_info:course_start_date', '01/23/2018'

```
hbase(main):001:0>
hbase(main):002:0* create 'user_courses', 'user_info', 'course_info'
0 row(s) in 1.9190 seconds

=> Hbase::Table - user_courses
hbase(main):003:0>
hbase(main):004:0* put 'user_courses', '1', 'user_info:user_id', '2341'
0 row(s) in 0.5100 seconds

hbase(main):005:0> put 'user_courses', '1', 'user_info:user_name', 'Avni Mehta'
0 row(s) in 0.0210 seconds

hbase(main):006:0>
hbase(main):007:0* put 'user_courses', '1', 'course_info:course_id', '21'
0 row(s) in 0.0220 seconds

hbase(main):008:0> put 'user_courses', '1', 'course_info:course_name', 'Python'
0 row(s) in 0.0130 seconds

hbase(main):009:0> put 'user_courses', '1', 'course_info:course_completion_status', 'In Progress'
0 row(s) in 0.0130 seconds

hbase(main):010:0> put 'user_courses', '1', 'course_info:course_start_date', '04/23/2018'
0 row(s) in 0.0110 seconds

hbase(main):011:0>
hbase(main):012:0* put 'user_courses', '2', 'user_info:user_id', '2341'
0 row(s) in 0.0140 seconds

hbase(main):013:0> put 'user_courses', '2', 'user_info:user_name', 'Avni Mehta'
0 row(s) in 0.0220 seconds

hbase(main):014:0>
hbase(main):015:0* put 'user_courses', '2', 'course_info:course_id', '341'
0 row(s) in 0.0040 seconds

hbase(main):016:0> put 'user_courses', '2', 'course_info:course_name', 'AWS'
0 row(s) in 0.0120 seconds

hbase(main):017:0> put 'user_courses', '2', 'course_info:course_completion_status', 'Complete'
0 row(s) in 0.0090 seconds

hbase(main):018:0> put 'user_courses', '2', 'course_info:course_start_date', '06/18/2018'
0 row(s) in 0.0060 seconds

hbase(main):019:0>
hbase(main):020:0* put 'user_courses', '3', 'user_info:user_id', '10'
0 row(s) in 0.0080 seconds
```

HBase queries:

Scan all rows of table
scan 'user_courses'

```
hbase(main):029:0> scan 'user_courses'
ROW                                                   COLUMN+CELL
 1                                                    column=course_info:course_completion_status, timestamp=1529368977311, value=In Progress
 1                                                    column=course_info:course_id, timestamp=1529368977144, value=21
 1                                                    column=course_info:course_name, timestamp=1529368977235, value=Python
 1                                                    column=course_info:course_start_date, timestamp=1529368977393, value=04/23/2018
 1                                                    column=user_info:user_id, timestamp=1529368976735, value=2341
 1                                                    column=user_info:user_name, timestamp=1529368976904, value=Avni Mehta
 2                                                    column=course_info:course_completion_status, timestamp=1529368977786, value=Complete
 2                                                    column=course_info:course_id, timestamp=1529368977709, value=341
 2                                                    column=course_info:course_name, timestamp=1529368977745, value=AWS
 2                                                    column=course_info:course_start_date, timestamp=1529368977842, value=06/18/2018
 2                                                    column=user_info:user_id, timestamp=1529368977495, value=2341
 2                                                    column=user_info:user_name, timestamp=1529368977583, value=Avni Mehta
 3                                                    column=course_info:course_completion_status, timestamp=1529368978141, value=Complete
 3                                                    column=course_info:course_id, timestamp=1529368978066, value=21
 3                                                    column=course_info:course_name, timestamp=1529368978113, value=Python
 3                                                    column=course_info:course_start_date, timestamp=1529368979523, value=01/23/2018
 3                                                    column=user_info:user_id, timestamp=1529368977889, value=10
 3                                                    column=user_info:user_name, timestamp=1529368977982, value=Hardik Mehta
3 row(s) in 0.2230 seconds
```

Get all rows in a specific a timerange (eg. courses enrolled today)
scan 'user_courses', {TIMERANGE => [1303668804, 1303668904]}

```
hbase(main):001:0> scan 'user_courses', {TIMERANGE => [1529368977311, 1539368977311]}
ROW                                                   COLUMN+CELL
 1                                                    column=course_info:course_completion_status, timestamp=1529368977311, value=In Progress
 1                                                    column=course_info:course_start_date, timestamp=1529368977393, value=04/23/2018
 2                                                    column=course_info:course_completion_status, timestamp=1529368977786, value=Complete
 2                                                    column=course_info:course_id, timestamp=1529368977709, value=341
 2                                                    column=course_info:course_name, timestamp=1529368977745, value=AWS
 2                                                    column=course_info:course_start_date, timestamp=1529368977842, value=06/18/2018
 2                                                    column=user_info:user_id, timestamp=1529368977495, value=2341
 2                                                    column=user_info:user_name, timestamp=1529368977583, value=Avni Mehta
```

Get user name and course name
scan 'user_courses', {COLUMNS => ['user_name', 'course_name']}

```
hbase(main):007:0> scan 'user_courses', {COLUMNS => ['user_info']}
ROW                                                   COLUMN+CELL
 1                                                    column=user_info:user_id, timestamp=1529368976735, value=2341
 1                                                    column=user_info:user_name, timestamp=1529368976904, value=Avni Mehta
 2                                                    column=user_info:user_id, timestamp=1529368977495, value=2341
 2                                                    column=user_info:user_name, timestamp=1529368977583, value=Avni Mehta
 3                                                    column=user_info:user_id, timestamp=1529368977889, value=10
 3                                                    column=user_info:user_name, timestamp=1529368977982, value=Hardik Mehta
3 row(s) in 0.0820 seconds
```

c) Use Cassandra to implement a Solution for the use case. Report at least 3 queries, their input and output. The query's relevance towards solving the use case is important.

<u>Table Creation</u>

```
create keyspace coursera with replication={'class':'SimpleStrategy', 'replication_factor':1};

USE coursera;
CREATE TABLE user_courses (
        user_id int,
        user_name text,
        course_id int,
        course_name text,
        course_completion_status text,
        course_start_date timestamp,
        PRIMARY KEY(user_id, course_id)
);
```

```
cqlsh>
cqlsh> create keyspace coursera with replication={'class':'SimpleStrategy', 'rep
lication_factor':1};
cqlsh>
cqlsh> USE coursera;
cqlsh:coursera>
cqlsh:coursera> CREATE TABLE user_courses ( user_id int, user_name text, course_
id int, course_name text, course_completion_status text, course_start_date times
tamp, PRIMARY KEY (user_id, course_id));
cqlsh:coursera>
cqlsh:coursera> select * from user_courses;

 user_id | course_id | course_completion_status | course_name | course_start_dat
e | user_name
---------+-----------+--------------------------+-------------+-----------------
--+-----------

(0 rows)
cqlsh:coursera>
```

Insert Records

```
INSERT INTO user_courses (user_id, user_name, course_id, course_name, course_completion_status, course_start_date)
  VALUES (2341, 'Avni Mehta', 21, 'Python', 'In Progress', '2018-02-05');

INSERT INTO user_courses (user_id, user_name, course_id, course_name, course_completion_status, course_start_date)
  VALUES (2341, 'Avni Mehta', 15, 'AWS', 'Complete', '2018-05-05');

INSERT INTO user_courses (user_id, user_name, course_id, course_name, course_completion_status, course_start_date)
  VALUES (10, 'Hardik Mehta', 21, 'Python', 'Complete', '2017-05-01');
```

```
cqlsh:coursera> INSERT INTO user_courses (user_id, user_name, course_id, course_
name, course_completion_status, course_start_date) VALUES (2341, 'Avni Mehta', 2
1, 'Python', 'In Progress', '2018-02-05');
cqlsh:coursera>
cqlsh:coursera> INSERT INTO user_courses (user_id, user_name, course_id, course_
name, course_completion_status, course_start_date) VALUES (2341, 'Avni Mehta', 1
5, 'AWS', 'Complete', '2018-05-05');
cqlsh:coursera>
cqlsh:coursera> INSERT INTO user_courses (user_id, user_name, course_id, course_
name, course_completion_status, course_start_date) VALUES (10, 'Hardik Mehta', 2
1, 'Python', 'Complete', '2017-02-05');
cqlsh:coursera>
```

Cassandra Queries

Scan all rows of table

```
select * from user_courses
```

```
cqlsh:coursera> select * from user_courses;

 user_id | course_id | course_completion_status | course_name | course_start_dat
e        | user_name
---------+-----------+--------------------------+-------------+-----------------
---------+--------------
    2341 |        15 |                 Complete |         AWS | 2018-05-05 04:00
:00+0000 |   Avni Mehta
    2341 |        21 |              In Progress |      Python | 2018-02-05 05:00
:00+0000 |   Avni Mehta
      10 |        21 |                 Complete |      Python | 2017-02-05 05:00
:00+0000 | Hardik Mehta

(3 rows)
cqlsh:coursera>
```

Get all rows in a specific a timerange

```
select * from user_courses where course_start_date > '2017-05-01' ALLOW FILTERING;
```

```
cqlsh:coursera> select * from user_courses where course_start_date > '2017-03-01
' ALLOW FILTERING;

 user_id | course_id | course_completion_status | course_name | course_start_dat
e        | user_name
---------+-----------+--------------------------+-------------+-----------------
---------+------------
    2341 |        15 |                 Complete |         AWS | 2018-05-05 04:00
:00+0000 | Avni Mehta
    2341 |        21 |              In Progress |      Python | 2018-02-05 05:00
:00+0000 | Avni Mehta

(2 rows)
```

Get user name and course name
select user_name, course_name from user_courses;

```
cqlsh:coursera>
cqlsh:coursera> select user_name, course_name from user_courses;

 user_name    | course_name
--------------+-------------
   Avni Mehta |         AWS
   Avni Mehta |      Python
 Hardik Mehta |      Python

(3 rows)
```

Demo Video:
Video for the demo can be found at https://youtu.be/jPJVIpPsUOM