BIGDATA PROGRAMMING

Lab1 Assignment

Jetty, Lalitha Kumari – Class ID 5
Vinay Chandra Vasamsetti – Class ID 25
Team 11

1. Hadoop MapReduce Algorithm

Implement MapReduce algorithm for finding Facebook common friends problem and run the MapReduce job on Apache Hadoop. Show your implementation through map-reduce diagram as shown in Lesson Plan 2 : (https://umkc.box.com/s/epp04s3m6g8jw6ulnnb4meqgnbwbz5uc) Write a report including your algorithm and result screenshots.

Finding Facebook common friends: Facebook has a list of friends (note that friends are a bi-directional thing on Facebook. If I'm your friend, you're mine). They also have lots of disk space and they serve hundreds of millions of requests everyday. They've decided to pre-compute calculations when they can to reduce the processing time of requests. One common processing request is the "You and Joe have 230 friends in common" feature. When you visit someone's profile, you see a list of friends that you have in common. We're going to use MapReduce so that we can calculate everyone's common friends once a day and store those results. Later on it's just a quick lookup. We've got lots of disk, it's cheap.

Example (What is the Key/Value Pair?)

Assume the friends are stored as Person->[List of Friends], our friends list is then:

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

The result after reduction is:

(A B) -> (C D)
(A C) -> (B D)
(A D) -> (B C)
(B C) -> (A D E)
(B D) -> (A C E)
(B E) -> (C D)
(C D) -> (A B E)
(C E) -> (B D)
(D E) -> (B C)

When D visits B's profile, we can quickly look up (B D) and see that they have three friends in common, (A C E).

Input		Mapper				Reducer		Output
		Key Value Pair						
A : B C D		A : B C D map => (A, B) => B C D (A, C) => B C D (A, D) => B C D		(A B) -> (A C D E) (B C D) (A C) -> (A B D E) (B C D) (A D) -> (A B C E) (B C D)		A B : C D A C : B D A D : B C		A B: C D A C: B D A D: B C B C: A D E B D: A C E B E: C D C D: A B E C E: B D D E: B C
B : A C D E		(A B) -> A C D E (B C) -> A C D E (B D) -> A C D E (B E) -> A C D E		(B C) -> (A B D E) (A C D E) (B D) -> (A B C E) (A C D E) . .		. . .		
C : A B D E		(A C) -> A B D E (B C) -> A B D E (C D) -> A B D E (C E) -> A B D E		. . .				
D : A B C E		(A D) -> A B C E (B D) -> A B C E (C D) -> A B C E (D E) -> A B C E		. .				
E : B C D		(B E) -> B C D (C E) -> B C D (D E) -> B C D		. .				

Input

A **: B C D**
B **: A C D E**
C **: A B D E**
D **: A B C E**
E **: B C D**

Mapper
Generates key value pairs for each user to its friends.

A **: B C D**
map =>
(A**,** B**)** => B C D
(A**,** C**)** => B C D
(A**,** D**)** => B C D

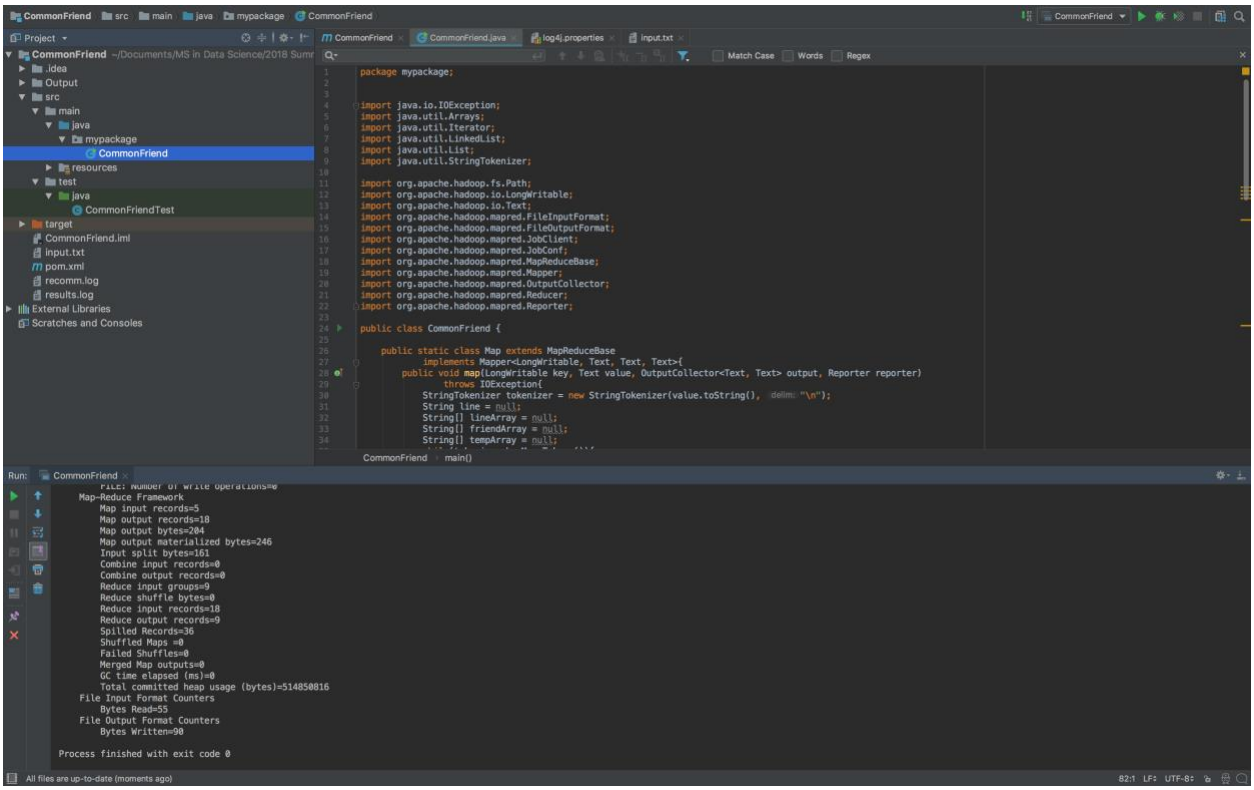
Reducer
Finds the common friends to the each user from the keyvalue pairs.

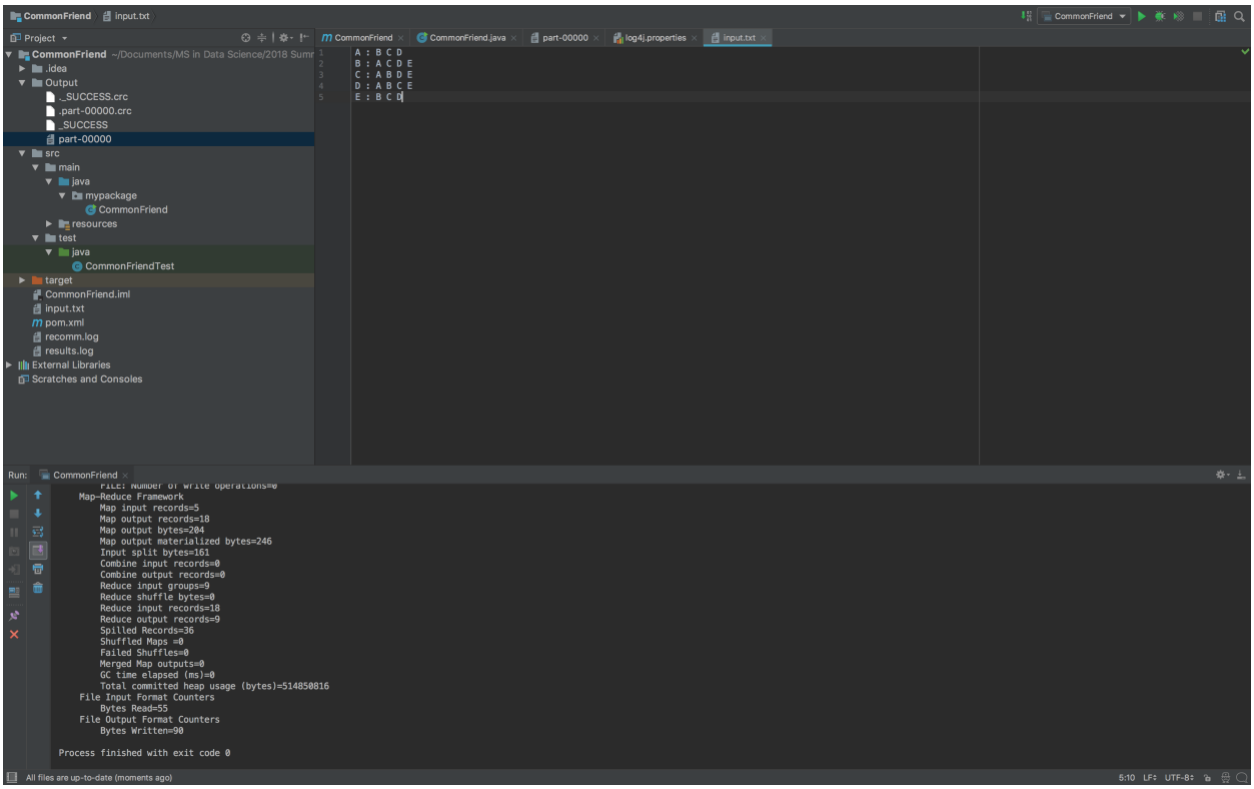
(A, B) => [B C D, A C D E]
reduce =>
(A, B) => C D

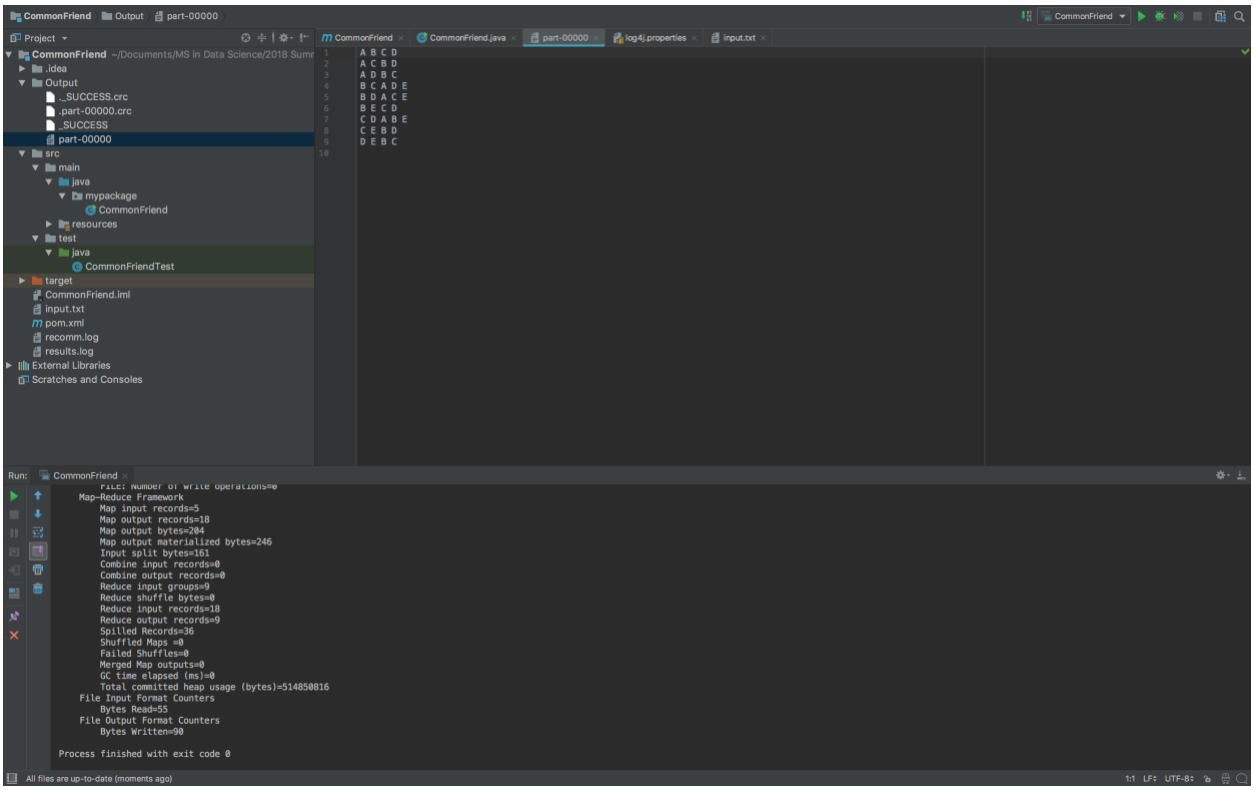
Output

A B: **C D**
A C: **B D**
A D: **B C**
B C: **A D E**
B D: **A C E**
B E: **C D**
C D: **A B E**
C E: **B D**
D E: **B C**

Code Screenshots







Reference - http://ernie55ernie.github.io/mapreduce/2016/06/30/map-reduce-common-friend.html

2. Use Case Based No SQL Comparison

Consider one of the use cases from the below link:
https://umkc.box.com/s/q64fvjm6yd454w5v3ky0he4854g6m1fq

These use cases were discussed in Lecture 1: Cassandra.

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.
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.
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.
d) Compare Cassandra and HBase for your use case. Present a table with comparison of your use case being implemented in both NO SQL Systems.

Use Case – Coursera

Description – Coursera is an Education Platform which partners with top universities and organizations worldwide, to offer courses online for anyone to take, for free.

Challenges –

1. My SQL was insufficient
2. Unstable performance
3. Unexpected downtime
4. Limitation in introducing new features

Solution – After evaluating emerging database technologies, it chooses Cassandra (data stax)

Reason – 100% application uptime needed and scalability (enabling storage of growing user data)

Creating Table in Hbase – Courses by Learner with column families as LearnerDetails, CourseDetails.

Commands Used:

```
create 'Coursera', 'LearnerDetails','CourseDetails'

put 'Coursera','1','LearnerDetails:LearnerID','111'
```

put 'Coursera', '1', 'LearnerDetails:LearnerName', 'Lalitha'

put 'Coursera', '2', 'LearnerDetails:LearnerID', '222'

put 'Coursera', '2, 'LearnerDetails:LearnerName', 'Vinay'

put 'Coursera', '1', 'CourseDetails:CourseID' ,'ML001'

put 'Coursera', '1', 'CourseDetails:CourseName' ,'Machine Learning'

put 'Coursera', '2, 'CourseDetails:CourseID' ,'DL001'

put 'Coursera', '2', 'CourseDetails:CourseName' ,'Deep Learning'

scan 'Coursera'

List

get 'Coursera', '1'

Creating Table in Cassandra – employee with ename, job title and hiredate

Creating Keyspace name Vinay

```
WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starti

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.2.8 | CQL spec 3.3.1 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh> create keyspace VINAY with replication={'class':'SimpleStrategy', 'replication_factor':1};
AlreadyExists: Keyspace 'vinay' already exists
cqlsh> use vinay;
```

Creating Table employee1

```
cqlsh:vinay> CREATE TABLE employee1 (
       ... empno int,
       ... ename text,
       ... jobtitle text,
       ... hiredate date,
       ... PRIMARY KEY (empno));
```

Inserting values in to table

```
SyntaxException: line 1:79 mismatched input '-06' expecting ')' (...) VALUES (1225,'katre',1993[-06]-12...)
cqlsh:vinay> INSERT INTO employee1 (empno,ename,hiredate,jobtitle) VALUES (1225,'katre','1993-06-12','softengin');
cqlsh:vinay> INSERT INTO employee1 (empno,ename,hiredate,jobtitle) VALUES (1245,'mourya','1980-12-17','manager');
cqlsh:vinay> INSERT INTO employee1 (empno,ename,hiredate,jobtitle) VALUES (1265,'santosh','1980-12-17','associate');
cqlsh:vinay> INSERT INTO employee1 (empno,ename,hiredate,jobtitle) VALUES (1265,'santosh','1980-12-17','associate');
cqlsh:vinay> SELECT * FROM employee1;

 empno | ename   | hiredate   | jobtitle
-------+---------+------------+-----------
  1245 |  mourya | 1980-12-17 |   manager
  1225 |   katre | 1993-06-12 | softengin
  1265 | santosh | 1980-12-17 | associate
```

Adding Column using Alter command

```
cqlsh:vinay> ALTER TABLE employee1 ADD salary int ;
cqlsh:vinay> SELECT * FROM employee1;

 empno | ename   | hiredate   | jobtitle  | salary
-------+---------+------------+-----------+--------
  1245 |  mourya | 1980-12-17 |   manager |   null
  1225 |   katre | 1993-06-12 | softengin |   null
  1265 | santosh | 1980-12-17 | associate |   null

(3 rows)
```

Updating salaries if the employees

```
cqlsh:vinay> update employee1 set salary = 10000 where empno = 1245;
cqlsh:vinay> update employee1 set salary = 20000 where empno = 1225;
cqlsh:vinay> update employee1 set salary = 30000 where empno = 1265;
cqlsh:vinay> SELECT * FROM employee1;

 empno | ename   | hiredate   | jobtitle  | salary
-------+---------+------------+-----------+--------
  1245 |  mourya | 1980-12-17 |   manager |  10000
  1225 |   katre | 1993-06-12 | softengin |  20000
  1265 | santosh | 1980-12-17 | associate |  30000
```

List name salary of the employee who are clerks

```
cqlsh:vinay> INDEX employee1 (jobtitle);
SyntaxException: line 1:0 no viable alternative at input 'INDEX' ([INDEX]...)
cqlsh:vinay> INDEX employee1(jobtitle);
SyntaxException: line 1:0 no viable alternative at input 'INDEX' ([INDEX]...)
cqlsh:vinay> CREATE INDEX on employee1 (jobtitle);
cqlsh:vinay> SELECT ename,salary from employee1
        ... WHERE(jobtitle = 'clerk');

 ename | salary
-------+--------
 sammy |   4000
```

List the name,job,salary for every employee joined on dec 17 1980

```
(1 rows)
cqlsh:vinay> CREATE INDEX on employee1 (hiredate);
cqlsh:vinay> SELECT * from employee1
        ... WHERE(hiredate = '1980-12-17');

 empno | ename  | hiredate   | jobtitle | salary
-------+--------+------------+----------+--------
  1245 | mourya | 1980-12-17 |  manager |  10000
  1265 |  sammy | 1980-12-17 |    clerk |   4000

(2 rows)
```

Similarities and differences between Cassandra and Hbase:

	Cassandra	**Hbase**
Database	open-source NoSQL Database, They both are distributed database that can manage extremely large data sets and handle non-relational data.	open-source NoSQL Database, They both are distributed database that can manage extremely large data sets and handle non-relational data.
Scalability	high linear scalability. That means to handle more data, the user should simply increase the number of nodes in the cluster. Because of this feature, they both are an excellent choice for handling a large amount of data.	high linear scalability. That means to handle more data, the user should simply increase the number of nodes in the cluster. Because of this feature, they both are an excellent choice for handling a large amount of data.

Replication	But in both Cassandra and HBase, there is a safeguard that prevents data loss even after failure	But in both Cassandra and HBase, there is a safeguard that prevents data loss even after failure
Programming/Coding	Both are column-oriented databases that implement similar write paths	Both are column-oriented databases that implement similar write paths
Infrastructure	Cassandra, on the other hand, has different infrastructure and operation than Hadoop	This HBase-Hadoop infrastructure consists of several moving parts like Zookeeper, HBase master, Data nodes and Name Node.
Support	Cassandra, on the other hand, supports ordered partitioning. Cassandra is also limited in supporting range based row scans.	HBase, do not supports ordered partitioning. HBase offers a coprocessor capability

Reference - https://data-flair.training/blogs/hbase-vs-cassandra/