

1번 문제.

HADOOP에서 HDFS에 읽었었는데, MONGODB파일을 HADOOP에서 불러오자.

## Start a MongoDB container

```
> docker run -d --name mongodb -p 27017:27017 mongo
```

Explain what does the -p option does? Is the option mandatory to run instructions in the next slides?

In the installed MongoDB container, load movies.csv, tags.csv

이게 무엇인가? 이 OPTION이 왜 필요한것인가

### 1번 ANSWER

DOCKER -P OPTION은 컨테이너 안에서 서비스를 구동하고, 외부에서 그 서비스에 접근하려고 할 때 사용한다. IP와 PORT로 그 컨테이너 안 서비스에 접속한다. 여기서는 27017:27017 中 첫 번째는 연결할 호스트의 포트 번호 이고, 뒤는 컨테이너의 포트 번호이다. 즉, -P 옵션은 컨테이너 포트를 열고 호스트 포트와 연결하는 옵션이다. 따라서 이 옵션을 사용하지 않았더라면, 뒤의 슬라이드를 진행하는데 어려움이 있다. 왜냐하면 MASTER NODE에서 컨테이너를 연결하고 실습을 진행하는 내용이기 때문이다.

## Download mongo-hadoop library from slave nodes

```
$ curl -fsSL \
https://raw.githubusercontent.com/kmu-bigdata/cloud-data-server/master/hadoop-mongo/download-hadoop-mongo-jar.sh | sudo sh
$ ls /usr/lib/hadoop-mapreduce/ | grep mongo-
```

Explain why the steps in page 10~12 are necessary.

```
kkvv111@khubigdata-clusters-w-1:~$ curl -fsSL https://raw.githubusercontent.com/kmu-bigdata/
cloud-data-server/master/hadoop-mongo/download-hadoop-mongo-jar.sh | sudo sh
--2018-10-16 01:24:24-- https://s3-us-west-2.amazonaws.com/cloud-data-server/mongo-hadoop-jar.zip
Resolving s3-us-west-2.amazonaws.com (s3-us-west-2.amazonaws.com)... 52.218.209.168
Connecting to s3-us-west-2.amazonaws.com (s3-us-west-2.amazonaws.com)[52.218.209.168]:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 1982451 (1.9M) [application/zip]
Saving to: 'mongo-hadoop-jar.zip'

mongo-hadoop-jar.zip  100%[=====] 1.89M  2.34MB/s  in 0.8s

2018-10-16 01:24:26 (2.34 MB/s) - 'mongo-hadoop-jar.zip' saved [1982451/1982451]

Archive: mongo-hadoop-jar.zip
  creating: mongo-hadoop-jar/
  inflating: mongo-hadoop-jar/mongo-java-driver-3.8.2.jar
  inflating: mongo-hadoop-jar/mongo-hadoop-core-2.0.2.jar
kkvv111@khubigdata-clusters-w-1:~$
kkvv111@khubigdata-clusters-w-1:~$
kkvv111@khubigdata-clusters-w-1:~$ ls /usr/lib/hadoop-mapreduce/ | grep mongo-
mongo-hadoop-core-2.0.2.jar
mongo-java-driver-3.8.2.jar
```

### 2번 ANSWER

10~12P의 내용들은 HOST(LOCAL)에서 MONGODB컨테이너에파일을 COPY하고 컨테이너에서는 MONGODB로 파일을 옮긴다. 그 후, 3개의 WORKER NODE에서 MONGOIMPORT를 하는데 여기서 MONGODB에 데이터를 넣는 것이다. 후자의 경우는 WORKER NODE 3개에 걸쳐 모두 진행한다. 컨테이너의 경우는 모든 NODE에서 접근할 수 있지만 MONGO에서 처리할 데이터의 경우는 각 NODE마다 줘야 하기 때문이다.

## MovieIdByTags Code

Explain what this code does

코드설

```
import com.mongodb.hadoop.MongoInputFormat;
import com.mongodb.hadoop.MongoOutputFormat;
import com.mongodb.hadoop.io.BSONWritable;
import com.mongodb.hadoop.util.MongoConfigUtil;

public class MovieIdByTags {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        MongoConfigUtil.setInputURI(conf, "mongodb://" + args[0]);
        MongoConfigUtil.setOutputURI(conf, "mongodb://" + args[1]);

        Job job = Job.getInstance(conf, "TagsAggregator");

        job.setJarByClass(MovieIdByTags.class);

        job.setMapperClass(Map.class);
        job.setCombinerClass(Combine.class);
        job.setReducerClass(Reduce.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(BSONWritable.class);

        job.setInputFormatClass(MongoInputFormat.class);
        job.setOutputFormatClass(MongoOutputFormat.class);

        job.waitForCompletion(true);
    }
}
```

### CAPTURE ANSWER

DRIVER CODE로서 각 단계의 입출력 형태를 지정한다. 이외에 기본적인 CONFIGURATION한다.

- ➔ MONGODB와 HADOOP간의 DATA를 주고 받기 위해서 CONFIGURATION을 설정하고 관련 API를 IMPORT한다. MR 작업을 위해 JOB 한 개를 생성한다. (WORDCOUNT의 경우는 JOB2개를 만들었다. 마지막에 SORT를 위한 MR작업을 위해) 그 후, MOVIEIDBYTAGS클래스를 바탕으로 하는 JAR파일을 만들도록 설정한다. 이후, MAPPER, COMBINER, REDUCE작업에 대한 클래스를 연결한다. 또한 각 단계별 INPUT, OUTPUT FORMAT을 설정한다.

전체 MAP의 INPUT, OUTPUT TYPE은 TEXT로 설정한다.

SETMAPOUTPUTKEYCLASS와 SETMAPOUTPUTVALUECLASS에서는 MAP의 출력에 대한 것인데, TEXT 형태로 KEY, VALUE 를 RETURN 할 것을 말한다. 원래는 SETOUTPUTKEYCLASS 등을 이용해서 한 번에 작성하지만, MAP과 REDUCE의 TYPE이 다르기 때문에 따로 설정한다.

또한 SETOUTPUTKEYCLASS, SETOUTPUTKEYVALUECLASS를 통해서는 최종 결과물에 대해서 TYPE을 정하는데 KEY, VALUE를 KEY는 TEXT이고 VALUE는 BSONWRITABLE로 이진 JSON TYPE으로, SERIALIZATION가능한 TYPE으로 보낸다는 것이다.

마지막으로 SETINPUTFORMATCLASS, SETOUTPUTFORMATCLASS를 정의하는데 이는, 전체 입출력 DATA FORMAT을 MONGO\_FORMAT으로 설정한다는 의미다.

## MoviedByTags Code

MoviedByTags Map

Explain what this code does

```
public static class Map extends Mapper<ObjectId, BSONObject, Text, Text> {
    private final Text tagOutput = new Text();
    private final Text movieIdOutput = new Text();

    public void map(ObjectId key, BSONObject value, Context context)
        throws IOException, InterruptedException {
        String tag = value.get("tag").toString();
        String movieId = value.get("movieId").toString();

        tagOutput.set(tag);
        movieIdOutput.set(movieId);
        context.write(tagOutput, movieIdOutput);
    }
}
```

### CAPTURE ANSWER

→MAP에서는 MAP의 INPUT KEY, VALUE를 OBJECTID와 BSONOBJECT(BINARY JSON)TYPE으로 정하고 OUTPUT의 KEY, VALUE는 TEXT TYPE으로 정한다. 또한 변수를 설정하는데 TAGOUTPUT, MovieIdOutput 은 text type의 상수형태다. MAP으로 들어오는 KEY-VALUE 입력에 대해서 TAG는 VALUE에서 TAG를 가져온 값을 STRING형태로 저장한다. 또한 MOVIEID에 MOVIEID를 가져와서 STRING형태로 가져온다. 그리고 TAG값을 tagoutput변수에 저장한다. 또한 movieIdOutput변수에 movieId값을 저장한다. 그 후에 tagsoutput과 movieIdOutput을 저장한다. 즉, map의 결과로 return한다. 정리하자면 value로 오는 BSONObject에 대해서 tag와 movieId값을 각각 tagoutput과 movieIdoutput으로 저장한 후 key는 tagoutput value는 movieIdOutput을 저장하여 return한다. 이 때 둘의 type은 text type이다.

## MoviedByTags Code

Explain what Combiner does

```
public static class Combine extends Reducer<Text, Text, Text, Text> {
    private final Text combinerOutput = new Text();

    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {
        HashSet<String> ids = new HashSet<>();
        for (Text value : values) {
            ids.add(value.toString());
        }

        Iterator itr = ids.iterator();
        while (itr.hasNext()) {
            combinerOutput.set(itr.next().toString());
            context.write(key, combinerOutput);
        }
    }
}
```

### Capture Answer

→먼저, combine이 MR PROGRAMMING에 있어서 어떤 역할을 하는지 설명하겠다. 이는 Optional  
로서 Map의 결과를 Local Write하고 Reduce에 보내는 과정에 있어서 그 때 각 map의 결과를 합  
치는 것이다. 예를 들어서 고양이1, 고양이1, 강아지1 이라고 map의 결과가 나온다면, 고양이2 강  
아지1로 Map의 결과물들을 combine한다.

여기서의 Combine을 설명하자면, Input, Output의 key-value가 모두 Text type이다. 입력에 대해서  
설명하자면 TYPE은 TEXT이지만, VALUE는 TEXT ITERABLE이다. 즉, KEY별로 VALUE를 ITERABLE하  
게 가지고 있다. 이것에 대해 TRAVERSE하는데, KEY마다 VALUE를 HASH자료구조에 STRING으로  
바꾸어 저장한다. 또한 HASHSET에 대해서 ITERATOR를 이용해서 COMBINEROUTPUT변수에  
HASH에 저장된 값을 저장한다. 그리고 해당 KEY(TAG)와 결과를VALUE(MOVIEID)로 RETURN한다.

## MovieIdByTags Code

### MovieIdByTags Reduce

Explain what this code does

```
public static class Reduce extends Reducer<Text, Text, Text, BSONWritable> {
    private BSONWritable reduceResult = new BSONWritable();

    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {
        BasicBSONObject output = new BasicBSONObject();
        HashSet<String> ids = new HashSet<>();
        StringBuilder sb = new StringBuilder();

        for (Text value : values) {
            ids.add(value.toString());
        }

        Iterator itr = ids.iterator();
        while (itr.hasNext()) {
            sb.append(itr.next().toString());
            sb.append(", ");
        }

        String movieIds = sb.toString();
        if (movieIds.endsWith(", ")) {
            movieIds = movieIds.substring(0, movieIds.length() - 2);
        }

        output.put("movieIds", movieIds);
        reduceResult.setDoc(output);
        context.write(key, reduceResult);
    }
}
```

### CAPTURE ANSWER

→ 해당 코드는 REDUCE 과정이다. REDUCE의 INPUT가 OUTPUT KEY-VALUE는 INPUT의 경우는 COMBINE과정을 거친 것으로 KEY와 VALUE는 TEXT이다. 또한 OUTPUT의 KEY는 TEXT이고 VALUE는 BSONWRITABLE TYPE으로써 BINARY JSON TYPE이다. KEY(TAG)마다 가진 MOVIEID(ITERABLE MOVIEID)를 읽어서 HASHSET에 저장한다. (이 때, SET이므로 중복이 없다.) 그리고 HASHSET을 ITERATOR로 TRAVERSE하면서 STRINGBUILDER TYPE가진 SB변수에 STRING TYPE으로 MOVIEID를 APPEND해서 저장한다. 또한 MOVIEIDS라는 STRING에 MOVIEID를 APPEND해서 저장한 SB를 저장한다. 그 후, 해당 STRING TYPE의 MOVIEIDS가 "," 형태로 끝난다면 그 앞까지 잘라서 MOVIEIDS에 저장한다. 그 후, BASICBSONOBJECT에 "MOVIEIDS"이름으로 앞의 과정 결과를 저장한다. 그 후 BSONWRITABLE TYPE형식에 맞게 SETDOC함수를 통해 저장하고 REDUCE를 마친다.

결과, TAG별 MOVIEID 모음이 저장된다. (BSON TYPE을 MONGODB로 읽어서 출력해볼 수 있다)



## Capture the screen of MongoDB outputs of

*movielens.tags\_out table*

## -capture결과-

```
root@c714fed85635: / - Chrome
https://ssh.cloud.google.com/projects/sound-decoder-221702/zones/asia-northeast1-c/instances/kmubigdata-cluster-m?authuser=1&hl=ko&projectNumber=176045424343

lned :
@ (shell):1:1
> db.tags_out.find({})
{ "id": "1950's Superman TV show", "movieIds": "47950" }
{ "id": "\"The Hunter\"", "movieIds": "100108, 26172, 2490" }
{ "id": "\"Bad CIA\"-- too simplistic", "movieIds": "74545" }
{ "id": "\"Based on true events\"", "movieIds": "6746" }
{ "id": "\"bitchy\"", "movieIds": "111800, 926" }
{ "id": "\"singing by non-singers\" in The Nyer 24/31 Dec 2012", "movieIds": "99149" }
{ "id": "\"scifi", "movieIds": "120476" }
{ "id": "\"AKA: I Will Never Get Those 2 Hours Back...Ever\"", "movieIds": "5680" }
{ "id": "\"I Had No Idea That You Were A F***** Vampire!", "movieIds": "49961" }
{ "id": "\"It's Not Where Your Dreams Take You But Who You Take In Your Dreams' hahaha!", "movieIds": "2076" }
{ "id": "\"Show Me The Money! I Can't Hear You Haji Scream Louder!", "movieIds": "55284" }
{ "id": "\"Wow Mrs. Robinson Your Porridge Is Just Right!", "movieIds": "1247" }
{ "id": "\"(mostly) for kids", "movieIds": "5971" }
{ "id": "\"(s)vcd", "movieIds": "6157, 7369, 1286, 2078, 2297, 8493, 594, 596, 3 611, 2687, 215, 2761, 6743, 912, 8464, 2080, 5410, 4366, 3034, 27899, 6281, 2161, 8360, 588, 1025, 1024, 8638, 4973, 8636, 5349, 2716, 8959" }
{ "id": "+++++", "movieIds": "61240" }
{ "id": "...to pay me to watch this again.", "movieIds": "3082" }
{ "id": "03/10", "movieIds": "8332, 69685, 8133, 64993, 8777, 42681, 6434, 431 77, 73587, 70432, 25908, 74510, 26797, 26796, 48909, 25941, 49917, 2270, 74152, 6 151, 45837, 27329, 33893, 115, 26113, 37335, 48660, 51127, 68472, 74508, 69498, 6 8486, 6761, 49225, 71011, 63676, 32196, 67999, 25911, 52784, 1420, 26245, 32792, 55895, 48575, 50245, 52424, 4856, 27370, 6692, 72117, 47634, 8275, 8511, 41336, 7 181, 66317, 49817, 39818, 27067, 53883, 8748, 6206, 47152, 54419, 71438, 5130, 73 51, 8484, 42213, 66320, 7113, 25972, 26268, 26228, 71390, 39481, 7195, 65709, 408 28, 27671, 31247, 73290, 2930, 3303, 57972, 8954, 54426, 52528, 37626, 7949, 3192 1" }
{ "id": "04/11", "movieIds": "6075, 8777, 26797, 41712, 48909, 60984, 32280, 5 6370, 82667, 79533, 60990, 37976, 44655, 42351, 26049, 25751, 26602, 79760, 54241 , 31188, 59655, 48575, 4856, 27370, 53835, 47084, 8794, 6133, 47239, 61742, 41815 , 59549, 81910, 27067, 84187, 7113, 44597, 25972, 33340, 80648, 9011, 27515, 2775 8, 50583, 58223, 76158, 26025, 60384, 8954, 73608, 53737, 8057, 8611, 6151, 45837 , 26998, 50594, 79578, 26596, 60551, 77833, 34548, 74727, 72142, 47728, 27738, 27 736, 59976, 32352, 53355, 58808, 44568, 5520, 70015, 71189, 4431, 69949, 41336, 7 7293, 39818, 25881, 77854, 6688, 45891, 6206, 4668, 48638, 5130, 40591, 47146, 81 817, 26228, 34238, 31247, 60103, 33264, 57972, 78898, 60461, 7949, 7828" }
{ "id": "05.02.06", "movieIds": "27416" }
{ "id": "05/09", "movieIds": "63808, 8017, 7479, 26269, 51127, 67508" }
type "it" for more
>
{ "id": "\"AKA: I Will NEVER Get Those 2 Hours Back...Ever\"", "movieIds": "5680" }
{ "id": "\"I Had No Idea That You Were A F***** Vampire!", "movieIds": "49961" }
{ "id": "\"It's Not Where Your Dreams Take You But Who You Take In Your Dreams' hahaha!", "movieIds": "2076" }
{ "id": "\"Show Me The Money! I Can't Hear You Haji Scream Louder!", "movieIds": "55284" }
{ "id": "\"Wow Mrs. Robinson Your Porridge Is Just Right!", "movieIds": "1247" }
{ "id": "\"(mostly) for kids", "movieIds": "5971" }
{ "id": "\"(s)vcd", "movieIds": "6157, 7369, 1286, 2078, 2297, 8493, 594, 596, 3 611, 2687, 215, 2761, 6743, 912, 8464, 2080, 5410, 4366, 3034, 27899, 6281, 2161, 8360, 588, 1025, 1024, 8638, 4973, 8636, 5349, 2716, 8959" }
{ "id": "+++++", "movieIds": "61240" }
{ "id": "...to pay me to watch this again.", "movieIds": "3082" }
{ "id": "03/10", "movieIds": "8332, 69685, 8133, 64993, 8777, 42681, 6434, 431 77, 73587, 70432, 25908, 74510, 26797, 26796, 48909, 25941, 49917, 2270, 74152, 6 151, 45837, 27329, 33893, 115, 26113, 37335, 48660, 51127, 68472, 74508, 69498, 6 8486, 6761, 49225, 71011, 63676, 32196, 67999, 25911, 52784, 1420, 26245, 32792, 55895, 48575, 50245, 52424, 4856, 27370, 6692, 72117, 47634, 8275, 8511, 41336, 7 181, 66317, 49817, 39818, 27067, 53883, 8748, 6206, 47152, 54419, 71438, 5130, 73 51, 8484, 42213, 66320, 7113, 25972, 26268, 26228, 71390, 39481, 7195, 65709, 408 28, 27671, 31247, 73290, 2930, 3303, 57972, 8954, 54426, 52528, 37626, 7949, 3192 1" }
{ "id": "04/11", "movieIds": "6075, 8777, 26797, 41712, 48909, 60904, 32280, 5 6370, 82667, 79533, 60990, 37976, 44655, 42351, 26049, 25751, 26602, 79760, 54241 , 31188, 59655, 48575, 4856, 27370, 53835, 47084, 8794, 6133, 47239, 61742, 41815 , 59549, 81910, 27067, 84187, 7113, 44597, 25972, 33340, 80648, 9011, 27515, 2775 8, 50583, 58223, 76158, 26025, 60384, 8954, 73608, 53737, 8057, 8611, 6151, 45837 , 26998, 50594, 79578, 26596, 60551, 77833, 34548, 74727, 72142, 47728, 27738, 27 736, 59976, 32352, 53355, 58808, 44568, 5520, 70015, 71189, 4431, 69949, 41336, 7 7293, 39818, 25881, 77854, 6688, 45891, 6206, 4668, 48638, 5130, 40591, 47146, 81 817, 26228, 34238, 31247, 60103, 33264, 57972, 78898, 60461, 7949, 7828" }
{ "id": "05.02.06", "movieIds": "27416" }
{ "id": "05/09", "movieIds": "63808, 8017, 7479, 26269, 51127, 67508" }
type "it" for more
> exit()
2018-11-21T09:20:05.275+0000 E QUERY [js] ReferenceError: exit is not defined :
@ (shell):1:1
> exit
bye
root@c714fed85635: /# pwd
/
root@c714fed85635: /# exit
exit
choirak0805@kmubigdata-cluster-m:~$ pwd
/home/choirak0805
choirak0805@kmubigdata-cluster-m:~$
```

PWD명령으로 HOST이름 최략존

**==EXERCISE==**

1. Using the movielens dataset, write a MongoDB query and Hive query to count the number of unique movie names (movies.csv)

**-HIVE-**

```
hive> select count(distinct title) from movies;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = choirak0805_20181122095055_d4a30479-3e2b-420a-8929-8d9d2f4a313b
Total jobs = 1
Launching Job 1 out of 1
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_1542770342773_0002, Tracking URL = http://kmubigdata-cluster-m:8088/proxy/application_1542770342773_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1542770342773_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-11-22 09:51:07,822 Stage-1 map = 0%, reduce = 0%
2018-11-22 09:51:17,317 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.45 sec
2018-11-22 09:51:24,660 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.87 sec
MapReduce Total cumulative CPU time: 5 seconds 870 msec
Ended Job = job_1542770342773_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.87 sec HDFS Read: 1385885 HDFS Write: 105
SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 870 msec
OK
27263
Time taken: 30.505 seconds, Fetched: 1 row(s)
hive>
```

**-MongoDB-**

```
local      0.000GB
movielens  0.025GB
> use movielens
switched to db movielens
> db.movies.distinct("title").length
27263
> RESULT
QUERY
```



***2. Implement Hadoop MapReduce code to count the number of unique movie names***

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import java.util.Arrays;
import java.util.ArrayList;

public class COUNT_UNIQUE_DRIVER {

    public static void main(String[] args) throws Exception{

        Configuration conf=new Configuration();

        Job job = Job.getInstance(conf,"COUNT_UNIQUE_DRIVER");

        job.setJarByClass(COUNT_UNIQUE_DRIVER.class);

        job.setNumReduceTasks(1); //To get unique title from multiple results from mapper

        job.setOutputKeyClass(Text.class); //KEY-VALUE (MOVIE_TITLE - COUNT)
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(COUNT_UNIQUE_MAP.class);
        job.setReducerClass(COUNT_UNIQUE_REDUCE.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job,new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path(args[1]));

        job.waitForCompletion(true);

    }
}
```

# DRIVER

```
}  
  
public static class COUNT_UNIQUE_MAP extends Mapper<LongWritable,Text,Text,IntWritable> {  
  
    private final static IntWritable ONE=new IntWritable(1);  
    private Text word=new Text();  
    ArrayList<String> titles=new ArrayList<String>();  
  
    public void map(LongWritable key,Text value,Context context)throws IOException,InterruptedException{  
        String line=value.toString();//READ LINE-BY-LINE  
        String [] fields=line.split(",");//DECODE -CSV  
        String title=fields[fields.length-2];//GET TITLE  
        //String[] titles=want_field;  
  
        boolean whether=true;  
  
        if(titles.contains(title)){  
            whether=false;  
        }  
  
        //Compare whether this title is saved in Word  
        if (whether==true) {  
            titles.add(title);  
            word.set(title);  
            context.write(word,ONE);//Return KEY-VALUE  
        }  
    }  
}
```

MAP

```
public static class COUNT_UNIQUE_REDUCE extends Reducer<Text,IntWritable,Text,IntWritable> {  
  
    /*  
    * REDUCE RETURN UNIQUE TITLE-ONE PER BLOCK. SO WE SHOULD GET UNIQUE IN REDUCE  
    */  
}
```

|          |           |          |            |          |            |           |            |              |           |               |
|----------|-----------|----------|------------|----------|------------|-----------|------------|--------------|-----------|---------------|
| Get Help | Write Out | Where Is | Cut Text   | Justify  | Cur Pos    | Prev Page | First Line | WhereIs Next | Mark Text | Indent Text   |
| Exit     | Read File | Replace  | Uncut Text | To Spell | Go To Line | Next Page | Last Line  | To Bracket   | Copy Text | Unindent Text |

```
    * REDUCE RETURN UNIQUE TITLE-ONE PER BLOCK. SO WE SHOULD GET UNIQUE IN REDUCE
    */
```

```
int for_res=0;
Text for_key= new Text();
```

```
public void reduce(Text key,Iterable<IntWritable> values,Context context) throws IOException,InterruptedException{
```

```
    int sum=0;
```

```
    for (IntWritable val:values) {
```

```
        sum+=val.get();
```

```
    }
```

```
    if (sum==1) {
```

```
        //context.write(key,new IntWritable (sum));
```

```
        for_res+=1;
```

```
    }
```

```
}
```

```
protected void cleanup(Context context) throws IOException, InterruptedException{
```

```
    for_key.set("");
```

```
    context.write(for_key,new IntWritable(for_res));
```

```
}
```

```
}
```

```
}
```

Get Help  
Exit

Write Out  
Read File

Where Is  
Replace

Cut Text  
Uncut Text

Justify  
To Spell

Cur Pos  
Go To Line

Prev Page  
Next Page

First Line  
Last Line

WhereIs Next  
To Bracket

Mark Text  
Copy Text

Indent Text  
Unindent Text

https://ssh.cloud.google.com/projects/sound-decoder-221702/zones/asia-northeast1-c/instances/kmubigdata-cluster-m?authuser=1&hl=ko&projectNumber=176045424343

```
HDFS: Number of write operations=3
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Rack-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=26576
  Total time spent by all reduces in occupied slots (ms)=13808
  Total time spent by all map tasks (ms)=6644
  Total time spent by all reduce tasks (ms)=3452
  Total vcore-milliseconds taken by all map tasks=6644
  Total vcore-milliseconds taken by all reduce tasks=3452
  Total megabyte-milliseconds taken by all map tasks=13606912
  Total megabyte-milliseconds taken by all reduce tasks=7069696
Map-Reduce Framework
  Map input records=27279
  Map output records=27263
  Map output bytes=891651
  Map output materialized bytes=946196
  Input split bytes=112
  Combine input records=0
  Combine output records=0
  Reduce input groups=27263
  Reduce shuffle bytes=946196
  Reduce input records=27263
  Reduce output records=1
  Spilled Records=54526
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=175
  CPU time spent (ms)=5520
  Physical memory (bytes) snapshot=717983744
  Virtual memory (bytes) snapshot=6973521920
  Total committed heap usage (bytes)=643301376
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=1377677
File Output Format Counters
  Bytes Written=8
choirak0805@kmubigdata-cluster-m:~/HW_03_MR/Q02$ hdfs dfs -cat /hw03/Q02/final_output/*
18/11/25 01:58:14 INFO gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.10-hadoop2
27263
```

**<--RESULT**

choirak0805@kmubigdata-cluster-m:~/HW\_03\_MR/Q02\$

RStudio

### 3. Using the movielens dataset, write a MongoDB query and Hive query to Count the number of movies whose genre contains a word "Film-Noir" (movies.csv)

#### -HIVE-

```
choirak0805@kmubigdata-cluster-m: ~/HW_03_MR/Q04 - Chrome
https://ssh.cloud.google.com/projects/sound-decoder-221702/zones/asia-northeast1-c/instances/kmubigdata-cluster-m?authuser=1&hl=ko&projectNumber=176045424343

hive> SELECT genre, COUNT(genre) AS genre_count FROM
> (SELECT explode(split(genres, '\\\\')) AS genre FROM movies)
> AS T2M2 GROUP BY genre;
QUERY
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = choirak0805_20181126161806_9e82ac52-99a0-4908-aac8-e532cc0bcc60
Total jobs = 1
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.reducers=<number>

> db.movies.aggregate([ {$project:{genre:{$split:["$genres",""]}}, {$match:{genre:"Film-Noir"}
},{$group:{_id:null,Film_Noir:{$sum:1}}},{$project:{_id:0}}]
{ db.movies.aggregate([ {$project:{genre:{$split:["$genres",""]}}, {$match:{genre:"Film-Noir"}},{$group:{_id:null,Film_Noir:{$sum:1}}},{$project:{_id:0}}]
{"Film_Noir": 330 }
ANSWER
330
Time taken: 26.08 seconds, Fetched: 1 row(s)
```

#### -MongoDB-

```
> db.movies.aggregate([ {$project:{genre:{$split:["$genres",""]}}, {$match:{genre:"Film-Noir"}
},{$group:{_id:null,Film_Noir:{$sum:1}}},{$project:{_id:0}}]
{ db.movies.aggregate([ {$project:{genre:{$split:["$genres",""]}}, {$match:{genre:"Film-Noir"}},{$group:{_id:null,Film_Noir:{$sum:1}}},{$project:{_id:0}}]
{"Film_Noir": 330 }
ANSWER
330
```

*4. Implement Hadoop MapReduce code to count the number of movies whose genre contains a word "Film-Noir" (movies.csv)*



```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import java.util.Arrays;
import java.util.ArrayList;

public class COUNT_FILM_NOIR {

    public static void main(String[] args) throws Exception{

        Configuration conf=new Configuration();

        Job job = Job.getInstance(conf,"COUNT_FILM_NOIR");

        job.setJarByClass(COUNT_FILM_NOIR.class);

        job.setNumReduceTasks(1); //To SUM "FILM-NOIR" GENRE TOGETHER AT ONE-REDUCE

        job.setOutputKeyClass(Text.class); //KEY-VALUE (MOVIE_TITLE - COUNT)
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(COUNT_FILM_NOIR_MAP.class);
        job.setReducerClass(COUNT_FILM_NOIR_REDUCE.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job,new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path(args[1]));

        job.waitForCompletion(true);
```

# DRIVER

```
        job.waitForCompletion(true);
    }

    public static class COUNT_FILM_NOIR_MAP extends Mapper<LongWritable,Text,Text,IntWritable> {

        private final static IntWritable ONE=new IntWritable(1);
        private Text word=new Text();

        public void map(LongWritable key,Text value,Context context)throws IOException,InterruptedException{
            String line=value.toString();//READ LINE-BY-LINE
            String [] fields=line.split(",");//DECODE -CSV
            String lastField=fields[fields.length-1];//GET GENRE
            String [] genres = lastField.split("\\(");
            //ONLY pass "FILM-NOIR"
            For (String genre:genres) {
                if(genre.equals("Film-Noir")) {
                    word.set(genre);
                    context.write(word,ONE);
                }
            }
        }
    }
}
```

MAP

```
public static class COUNT_FILM_NOIR_REDUCE extends Reducer<Text,IntWritable,Text,IntWritable> {

    /*
     * REDUCE RETURN UNIQUE TITLE-ONE PER BLOCK. SO WE SHOULD GET UNIQUE IN REDUCE
     */
    Text for key=new Text();
    int sum=0;

    public void reduce(Text key,Iterable<IntWritable> values,Context context) throws IOException,InterruptedException{
```

|          |           |          |            |          |            |           |            |              |           |               |
|----------|-----------|----------|------------|----------|------------|-----------|------------|--------------|-----------|---------------|
| Get Help | Write Out | Where Is | Cut Text   | Justify  | Cur Pos    | Prev Page | First Line | WhereIs Next | Mark Text | Indent Text   |
| Exit     | Read File | Replace  | Uncut Text | To Spell | Go To Line | Next Page | Last Line  | To Bracket   | Copy Text | Unindent Text |

```
public void reduce(Text key,Iterable<IntWritable> values,Context context) throws IOException,InterruptedException{  
    for_key.set(key);  
    for (IntWritable val:values) {  
        sum+=val.get();  
    }  
    protected void cleanup(Context context ) throws IOException ,InterruptedException{  
        context.write(for_key,new IntWritable(sum));  
        //context.write(key,new IntWritable (sum));  
    }  
}
```

# REDUCE

https://ssh.cloud.google.com/projects/sound-decoder-221702/zones/asia-northeast1-c/instances/kmubigdata-cluster-m?authuser=1&hl=ko&projectNumber=176045424343

```
HDFS: Number of write operations=3
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=16676
  Total time spent by all reduces in occupied slots (ms)=14208
  Total time spent by all map tasks (ms)=4169
  Total time spent by all reduce tasks (ms)=3552
  Total vcore-milliseconds taken by all map tasks=4169
  Total vcore-milliseconds taken by all reduce tasks=3552
  Total megabyte-milliseconds taken by all map tasks=8538112
  Total megabyte-milliseconds taken by all reduce tasks=7274496
Map-Reduce Framework
  Map input records=27279
  Map output records=330
  Map output bytes=4620
  Map output materialized bytes=5286
  Input split bytes=112
  Combine input records=0
  Combine output records=0
  Reduce input groups=1
  Reduce shuffle bytes=5286
  Reduce input records=330
  Reduce output records=1
  Spilled Records=660
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=172
  CPU time spent (ms)=1870
  Physical memory (bytes) snapshot=721809408
  Virtual memory (bytes) snapshot=6972743680
  Total committed heap usage (bytes)=642777088
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=1377677
File Output Format Counters
  Bytes Written=14
choirak0805@kmubigdata-cluster-m:~/HW_Q03_MR/Q04$ hdfs dfs -cat /hw03/Q04/final_final_output/*
18/11/25 01:47:45 INFO gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.10-hadoop2
Film-Noir      330
choirak0805@kmubigdata-cluster-m:~/HW_Q03_MR/Q04$
```

<- RESULT

5. Using the movielens dataset, write a MongoDB query and Hive query to list the tag that are cited the most (tags.csv)

-HIVE-

```
choirak0805@kmubigdata-cluster-m: ~/cloud-data-server/hive - Chrome
https://ssh.cloud.google.com/projects/sound-decoder-221702/zones/asia-northeast1-c/ins...
hive> select tag,count(*) as numoftag from tags group by tag order by numoftag desc limit 1;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = choirak0805_20181122100910_73064db4-dae4d09-8608-e5fa402166df
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_1542770342773_0004, Tracking URL = http://kmubigdata-cluster-m:8088/proxy/application_1542770342773_0004/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1542770342773_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-11-22 10:09:22,529 Stage-1 map = 0%, reduce = 0%
2018-11-22 10:09:29,991 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.06 sec
2018-11-22 10:09:36,254 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.53 sec
MapReduce Total cumulative CPU time: 6 seconds 530 msec
Ended Job = job_1542770342773_0004
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_1542770342773_0005, Tracking URL = http://kmubigdata-cluster-m:8088/proxy/application_1542770342773_0005/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1542770342773_0005
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-11-22 10:09:45,085 Stage-2 map = 0%, reduce = 0%
2018-11-22 10:09:51,429 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 2.6 sec
2018-11-22 10:09:57,678 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 4.2 sec
MapReduce Total cumulative CPU time: 4 seconds 200 msec
Ended Job = job_1542770342773_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.53 sec HDFS Read: 16611456 HDFS Write: 1318827 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 4.2 sec HDFS Read: 1324238 HDFS Write: 111 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 730 msec
OK
sci-fi 3384
Time taken: 48.293 seconds, Fetched: 1 row(s)
hive>
```

*-MongoDB-*

```
{ "Film_Noir" : 14 }  
> db.tags.aggregate([  
... {$group:{_id:"$tag",numoftag:{$sum:1}}},  
... {$sort:{numoftag:-1}},  
... {$limit:1}})  
{ "_id" : "sci-fi", "numoftag" : 3384 }  
> 20142770 ChoiRakJun To Certification
```

***6. Implement Hadoop MapReduce code to list the tag that are cited the most (tags.csv)***



```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import java.util.Arrays;
import java.util.ArrayList;

public class COUNT_MOST_CITED {

    public static void main(String[] args) throws Exception {

        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "COUNT_MOST_CITED");

        job.setJarByClass(COUNT_MOST_CITED.class);
        job.setNumReduceTasks(1); //To get unique title from multiple results from mapper

        // job.setNumReduceTasks(1); //To get unique title from multiple results from
        // mapper

        job.setOutputKeyClass(Text.class); // KEY-VALUE (MOVIE_TITLE ~ COUNT)
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(COUNT_MOST_CITED_MAP.class);
        job.setReducerClass(COUNT_MOST_CITED_REDUCE.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
```

# DRIVER

```
        job.waitForCompletion(true);
    }

    public static class COUNT_MOST_CITED_MAP extends Mapper<LongWritable, Text, Text, IntWritable> {

        private final static IntWritable ONE = new IntWritable(1);
        private Text word = new Text();

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String line = value.toString(); // READ LINE-BY-LINE
            String[] fields = line.split(","); // DECODE -CSV
            String tag = fields[fields.length - 2]; // GET TAG
            // String[] titles=want_field;

            word.set(tag);
            context.write(word, ONE);
        }
    }

    public static class COUNT_MOST_CITED_REDUCE extends Reducer<Text, IntWritable, Text, IntWritable> {

        int max = 0;
        Text maxWord = new Text();
        /*
         * REDUCE RETURN UNIQUE TITLE-ONE PER BLOCK. SO WE SHOULD GET UNIQUE IN REDUCE
         */

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

            int sum = 0;

            for (IntWritable val : values) {

                sum += val.get();

            } // SUM PER TAG
            if (sum > max) {
                max = sum;
                maxWord.set(key);
            }
        }
    }
}
```

MAP

REDUCE

```

        maxWord.set(key);
    }

    // context.write(key,new IntWritable (sum));
}

protected void cleanup(Context context )throws IOException, InterruptedException {
    context.write(maxWord, new IntWritable(max));
}
}
}

```

# REDUCE

^G Get Help    ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify    ^C Cur Pos    ^Y Prev Page    M-^ First Line    M-W WhereIs Next    ^^ Mark Text    M-} Indent Text  
 ^X Exit        ^R Read File    ^\_ Replace    ^U Uncut Text    ^T To Spell    ^G Go To Line    ^V Next Page    M-/ Last Line    M-] To Bracket    M-^ Copy Text    M-{ Unindent Text

https://ssh.cloud.google.com/projects/sound-decoder-221702/zones/asia-northeast1-c/instances/kmubigdata-cluster-m?authuser=1&hl=ko&projectNumber=176045424343

```
HDFS: Number of write operations=3
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=23964
  Total time spent by all reduces in occupied slots (ms)=17920
  Total time spent by all map tasks (ms)=5991
  Total time spent by all reduce tasks (ms)=4480
  Total vcore-milliseconds taken by all map tasks=5991
  Total vcore-milliseconds taken by all reduce tasks=4480
  Total megabyte-milliseconds taken by all map tasks=12269568
  Total megabyte-milliseconds taken by all reduce tasks=9175040
Map-Reduce Framework
  Map input records=465565
  Map output records=465565
  Map output bytes=7480657
  Map output materialized bytes=8411831
  Input split bytes=110
  Combine input records=0
  Combine output records=0
  Reduce input groups=38643
  Reduce shuffle bytes=8411831
  Reduce input records=465565
  Reduce output records=1
  Spilled Records=931130
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=214
  CPU time spent (ms)=5920
  Physical memory (bytes) snapshot=804356096
  Virtual memory (bytes) snapshot=6967902208
  Total committed heap usage (bytes)=717750272
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=16603996
File Output Format Counters
  Bytes Written=12
```

```
choirak0805@kmubigdata-cluster-m:~/HW_03_MR/Q06$ hdfs dfs -cat /hw03/Q06/final_output/*
18/11/24 16:05:27 INFO gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.10-hadoop2
sci-fi 3384
choirak0805@kmubigdata-cluster-m:~/HW_03_MR/Q06$
```

< RESULT