TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

LAB 1 - HADOOP MAPREDUCE

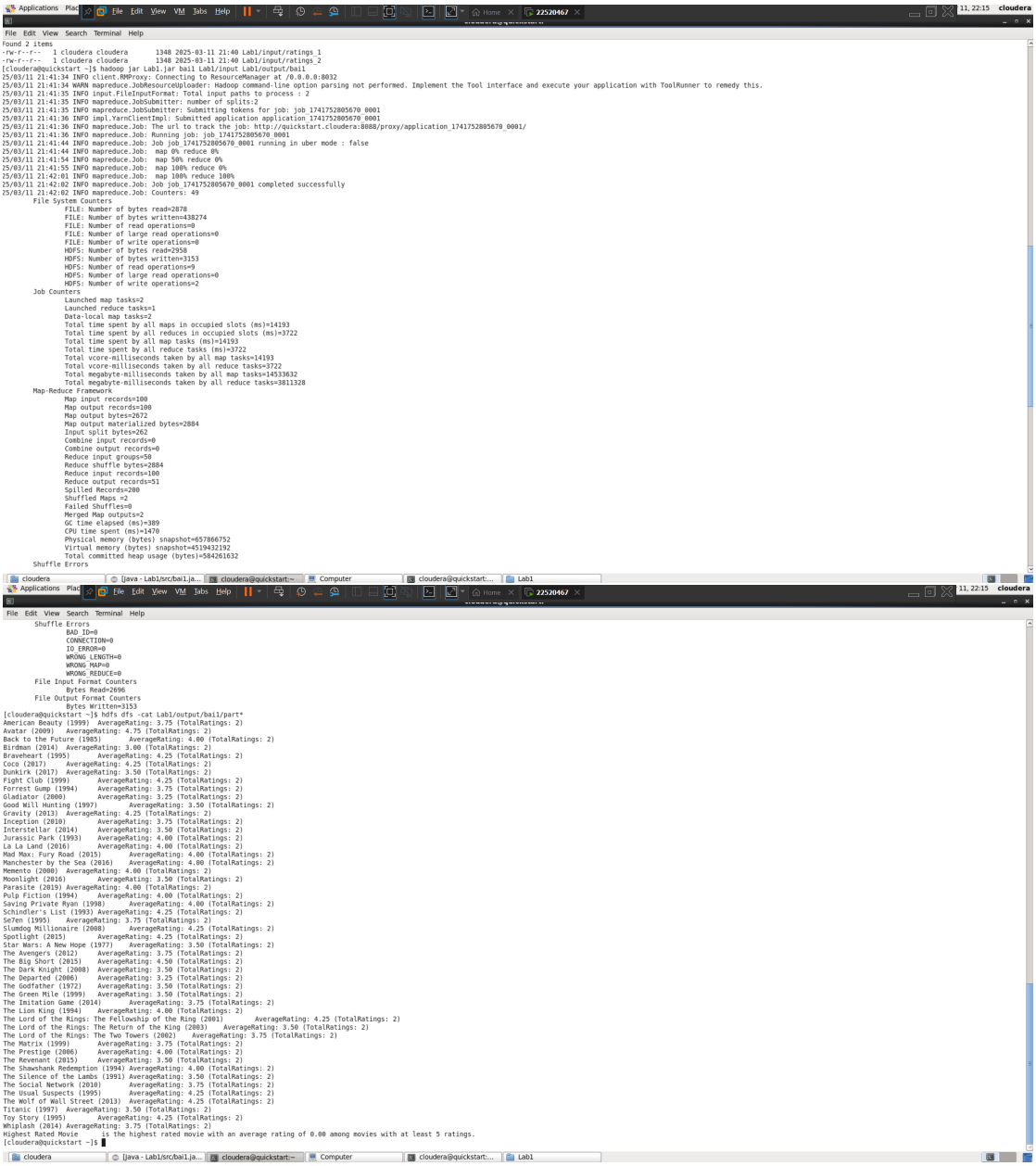
Lớp: DS200.P21

Họ và tên	**MSSV**
Nguyễn Duy Hoàng	22520467

Mục lục

1. Bài 1

1.1. Minh chứng



1.2. Code

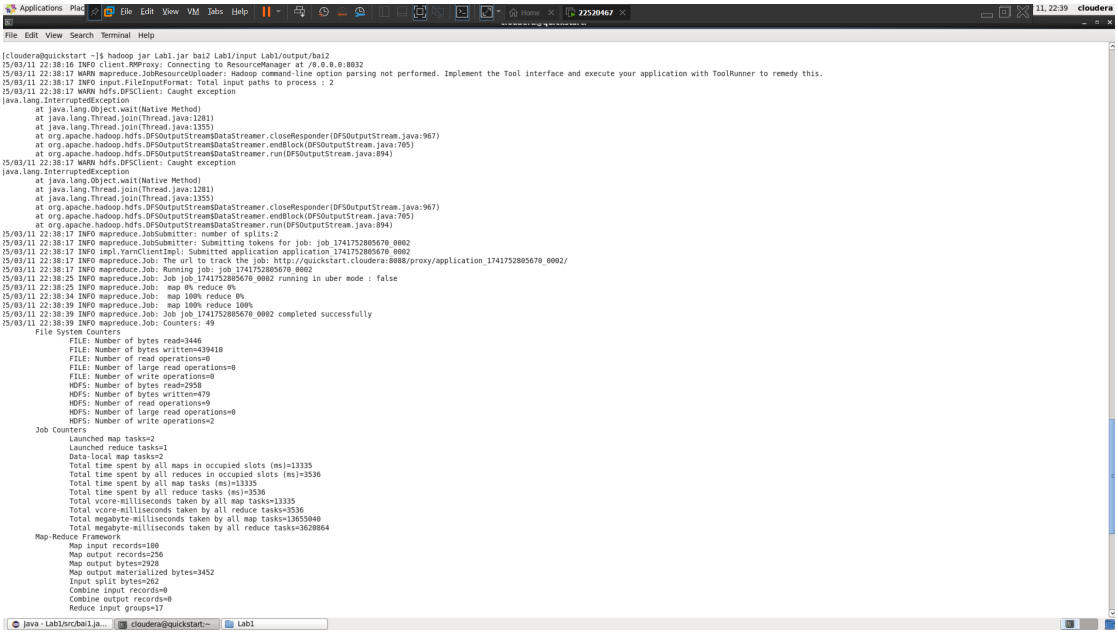
```java
// Import necessary libraries
import java.io.*;
import java.util.*;
import java.net.URI;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

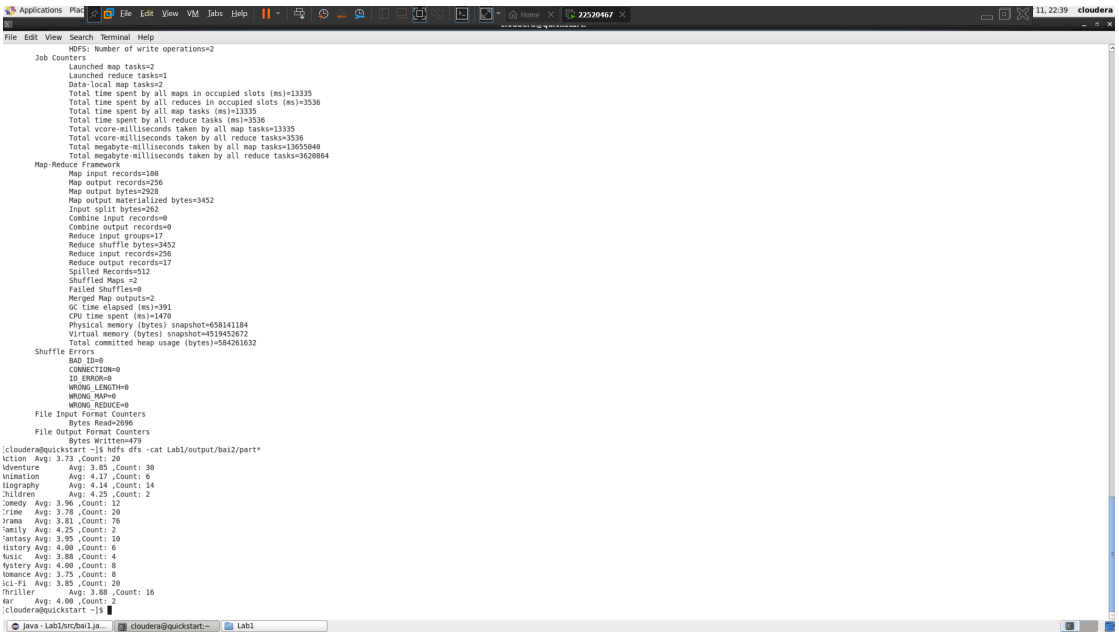
```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;


public class bai1 {

public static class RatingMapper extends Mapper <Object, Text, Text, Text>
{
        private Map<Integer, String> movieMap = new HashMap<>();

public void setup(Context context) throws IOException, InterruptedException
{
        try (BufferedReader br = new BufferedReader(new FileReader("movies"))) {
                String line;
                while ((line = br.readLine()) != null){
                        String columns[] = line.split(",");
                        String id = columns[0].trim();
                        String name = columns[1].trim();
                        movieMap.put(Integer.parseInt(id), name);
                }
        }
        catch (IOException e){
                e.printStackTrace();
        }
}

public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
        String record = value.toString().trim();
        String parts[] = record.split(",");

        String id_movie = parts[1].trim();
        String rating = parts[2].trim();
        String name_movie = movieMap.get(Integer.parseInt(id_movie));
        context.write(new Text(name_movie), new Text(rating));
}
}

public static class RatingReducer extends Reducer <Text, Text, Text, Text>
{
        String max_rating_movie = "";
        double max_rating = 0.0;

        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
        ↪  InterruptedException{
                double sum_rating = 0.0;
                int count_rating = 0;
                double temp_avg = 0.0;
            for (Text val : values) {
                sum_rating += Double.parseDouble(val.toString().trim());
                count_rating++;
            }
            if (count_rating >= 5){
                    temp_avg = (count_rating == 0) ? 0.0 : sum_rating / count_rating;
                    if (temp_avg > max_rating){
                            max_rating_movie = key.toString();
                            max_rating = temp_avg;
                    }
            }
            double avg = (count_rating == 0) ? 0.0 : sum_rating / count_rating;

            context.write(key, new Text(String.format("AverageRating: %.2f (TotalRatings: %d)",
            ↪  avg, count_rating)));
        }

    public void cleanup(Context context) throws IOException, InterruptedException {
            context.write(new Text("Highest Rated Movie"), new Text(String.format("%s is the
            ↪  highest rated movie with an average rating of %.2f among movies with at least 5
            ↪  ratings.", max_rating_movie, max_rating)));
    }

}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = new Job(conf, "Bai 1");
job.setJarByClass(bai1.class);
job.setMapperClass(RatingMapper.class);
```

```
job.setReducerClass(RatingReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
// Setting reducer to zero
//job.setNumReduceTasks(0);


try {

    job.addCacheFile(new URI("hdfs://localhost:8020/mycache/Lab1/movies"));
}
catch (Exception e) {
    System.out.println("File Not Added");
    System.exit(1);
}

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

2. Bài 2

2.1. Minh chứng





2.2. Code

```
import java.io.*;
import java.util.*;
import java.net.URI;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;


public class bai2 {

public static class RatingMapper extends Mapper <Object, Text, Text, Text>
{
        private Map<Integer, String> movieMap = new HashMap<>();

public void setup(Context context) throws IOException, InterruptedException
{
        try (BufferedReader br = new BufferedReader(new FileReader("movies"))) {
                String line;
                while ((line = br.readLine()) != null){
                        String columns[] = line.split(",");
                        String id = columns[0].trim();
                        String genres = columns[2].trim();
                        movieMap.put(Integer.parseInt(id), genres);
                }
        }
        catch (IOException e){
                e.printStackTrace();
        }
}

public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
        String record = value.toString().trim();
        String parts[] = record.split(",");

        String id_movie = parts[1].trim();
        String rating = parts[2].trim();
```

```
        String genres = movieMap.get(Integer.parseInt(id_movie));

    for (String genre : genres.split("\\|")) {
        context.write(new Text(genre.trim()), new Text(rating));
    }
}
}

public static class RatingReducer extends Reducer <Text, Text, Text, Text>
{
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
        ↪  InterruptedException{
                double sum_rating = 0.0;
                int count_rating = 0;
            for (Text val : values) {
                sum_rating += Double.parseDouble(val.toString().trim());
                count_rating++;
            }

            double avg = (count_rating == 0) ? 0.0 : sum_rating / count_rating;

            context.write(key, new Text(String.format("Avg: %.2f ,Count: %d", avg,
            ↪  count_rating)));
        }

}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = new Job(conf, "Bai 2");
job.setJarByClass(bai2.class);
job.setMapperClass(RatingMapper.class);
job.setReducerClass(RatingReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
// Setting reducer to zero
//job.setNumReduceTasks(0);


try {

    job.addCacheFile(new URI("hdfs://localhost:8020/mycache/Lab1/movies"));
}
catch (Exception e) {
    System.out.println("File Not Added");
    System.exit(1);
}

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

3. Bài 3

3.1. Minh chứng

```
[cloudera@quickstart ~]$ hadoop jar Lab1.jar bai3 Lab1/input Lab1/output/bai3
25/03/11 22:59:19 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
25/03/11 22:59:20 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
25/03/11 22:59:20 INFO input.FileInputFormat: Total input paths to process : 2
25/03/11 22:59:20 INFO mapreduce.JobSubmitter: number of splits:2
25/03/11 22:59:20 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1741752805670_0003
25/03/11 22:59:21 INFO impl.YarnClientImpl: Submitted application application_1741752805670_0003
25/03/11 22:59:21 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1741752805670_0003/
25/03/11 22:59:21 INFO mapreduce.Job: Running job: job_1741752805670_0003
25/03/11 22:59:28 INFO mapreduce.Job: Job job_1741752805670_0003 running in uber mode : false
25/03/11 22:59:28 INFO mapreduce.Job:  map 0% reduce 0%
25/03/11 22:59:37 INFO mapreduce.Job:  map 50% reduce 0%
25/03/11 22:59:38 INFO mapreduce.Job:  map 100% reduce 0%
25/03/11 22:59:45 INFO mapreduce.Job:  map 100% reduce 100%
25/03/11 22:59:45 INFO mapreduce.Job: Job job_1741752805670_0003 completed successfully
25/03/11 22:59:45 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=3078
                FILE: Number of bytes written=439316
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2958
                HDFS: Number of bytes written=2386
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=14298
                Total time spent by all reduces in occupied slots (ms)=3882
                Total time spent by all map tasks (ms)=14298
                Total time spent by all reduce tasks (ms)=3882
                Total vcore-milliseconds taken by all map tasks=14298
                Total vcore-milliseconds taken by all reduce tasks=3882
                Total megabyte-milliseconds taken by all map tasks=14641152
                Total megabyte-milliseconds taken by all reduce tasks=3975168
        Map-Reduce Framework
                Map input records=100
                Map output records=100
                Map output bytes=2872
                Map output materialized bytes=3084
                Input split bytes=262
                Combine input records=0
                Combine output records=0
                Reduce input groups=50
                Reduce shuffle bytes=3084
                Reduce input records=100
                Reduce output records=50
                Spilled Records=200
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=418
                CPU time spent (ms)=1610
                Physical memory (bytes) snapshot=664588288
                Virtual memory (bytes) snapshot=4519432192
                Total committed heap usage (bytes)=584261632
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
```



```
                Total committed heap usage (bytes)=584261632
        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=2696
        File Output Format Counters
                Bytes Written=2386
[cloudera@quickstart ~]$ hdfs dfs -cat Lab1/output/bai3/part*
%merican Beauty (1999)  Male: 3.00 ,Female: 4.50
%vatar (2009)   Male: 5.00 ,Female: 4.50
3ack to the Future (1985)       Male: 4.00 ,Female: 4.00
3irdman (2014)  Male: 3.00 ,Female: 3.00
3raveheart (1995)       Male: 4.50 ,Female: 4.00
:oco (2017)     Male: 5.00 ,Female: 3.50
Junkirk (2017)  Male: 4.00 ,Female: 3.00
'ight Club (1999)       Male: 5.00 ,Female: 3.50
'orrest Gump (1994)     Male: 2.50 ,Female: 5.00
;ladiator (2000)        Male: 3.50 ,Female: 3.00
Good Will Hunting (1997)        Male: 3.00 ,Female: 4.00
Gravity (2013)  Male: 4.00 ,Female: 4.50
Inception (2010)        Male: 3.50 ,Female: 4.00
Interstellar (2014)     Male: 3.50 ,Female: 3.50
Jurassic Park (1993)    Male: 3.50 ,Female: 4.50
.a La Land (2016)       Male: 5.00 ,Female: 3.00
Mad Max: Fury Road (2015)       Male: 4.00 ,Female: 4.00
Manchester by the Sea (2016)    Male: 3.50 ,Female: 4.50
Memento (2000)  Male: 3.50 ,Female: 4.50
Moonlight (2016)        Male: 3.00 ,Female: 4.00
Parasite (2019) Male: 4.00 ,Female: 4.00
Pulp Fiction (1994)     Male: 4.50 ,Female: 3.50
Saving Private Ryan (1998)      Male: 4.00 ,Female: 4.00
Schindler's List (1993) Male: 4.00 ,Female: 4.50
Se7en (1995)    Male: 4.00 ,Female: 3.50
Slumdog Millionaire (2008)      Male: 4.50 ,Female: 4.00
Spotlight (2015)        Male: 4.50 ,Female: 4.00
Star Wars: A New Hope (1977)    Male: 4.00 ,Female: 3.00
The Avengers (2012)     Male: 3.50 ,Female: 4.00
The Big Short (2015)    Male: 5.00 ,Female: 4.00
The Dark Knight (2008)  Male: 4.00 ,Female: 3.00
The Departed (2006)     Male: 4.50 ,Female: 2.00
The Godfather (1972)    Male: 4.50 ,Female: 2.50
The Green Mile (1999)   Male: 3.00 ,Female: 4.00
The Imitation Game (2014)       Male: 4.50 ,Female: 3.00
The Lion King (1994)    Male: 4.50 ,Female: 3.50
The Lord of the Rings: The Fellowship of the Ring (2001)        Male: 4.50 ,Female: 4.00
The Lord of the Rings: The Return of the King (2003)    Male: 4.00 ,Female: 3.00
The Lord of the Rings: The Two Towers (2002)    Male: 4.50 ,Female: 3.00
The Matrix (1999)       Male: 3.00 ,Female: 4.50
The Prestige (2006)     Male: 4.00 ,Female: 4.00
The Revenant (2015)     Male: 3.50 ,Female: 3.50
The Shawshank Redemption (1994) Male: 3.00 ,Female: 5.00
The Silence of the Lambs (1991) Male: 3.50 ,Female: 3.50
The Social Network (2010)       Male: 3.50 ,Female: 4.00
The Usual Suspects (1995)       Male: 4.00 ,Female: 4.50
The Wolf of Wall Street (2013)  Male: 4.00 ,Female: 4.50
Titanic (1997)  Male: 4.00 ,Female: 3.00
Toy Story (1995)        Male: 4.50 ,Female: 4.00
Whiplash (2014) Male: 4.50 ,Female: 3.00
[cloudera@quickstart ~]$
```

3.2. Code

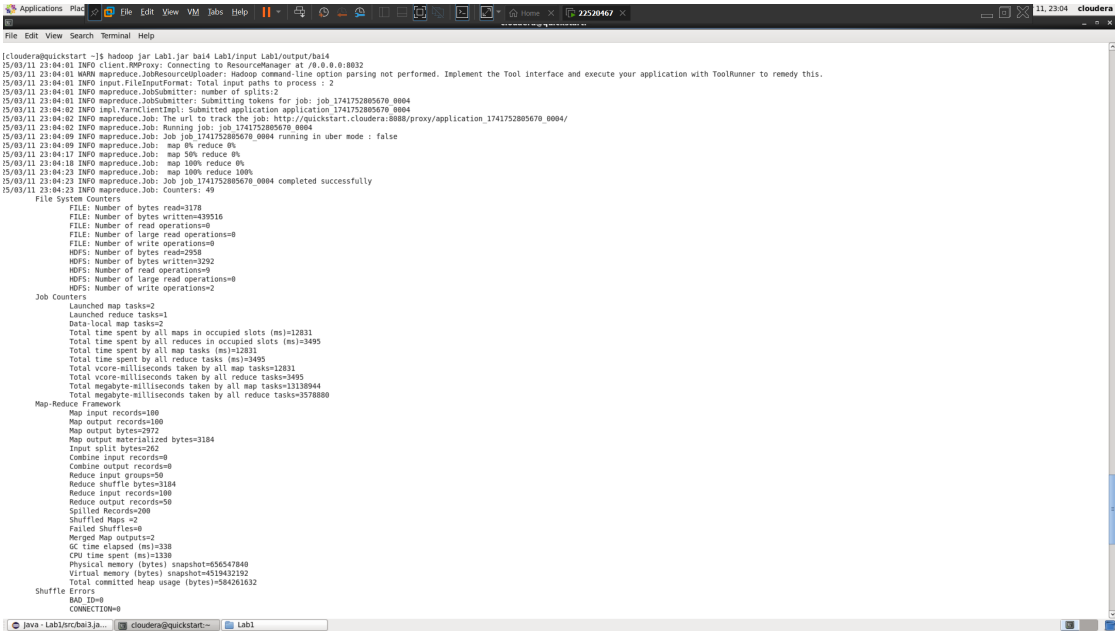
```
import java.io.*;
import java.util.*;
import java.net.URI;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.fs.Path;
import java.net.URI;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

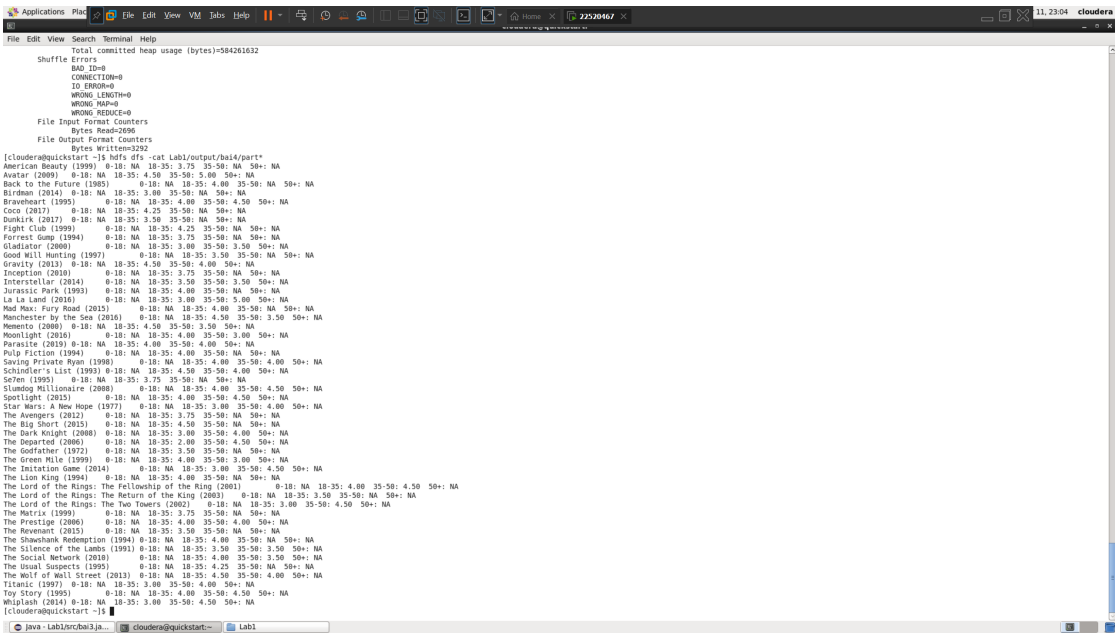
```
public class bai3 {

public static class RatingMapper extends Mapper <Object, Text, Text, Text>
{
        private Map<Integer, String> movieMap = new HashMap<>();
        private Map<Integer, String> userMap = new HashMap<>();

public void setup(Context context) throws IOException, InterruptedException
{
        URI[] CacheFiles = context.getCacheFiles();
        System.out.println("Number of cache files: " + (CacheFiles == null ? 0 :
        ↪  CacheFiles.length));

        if (CacheFiles != null && CacheFiles.length > 0){
                try (BufferedReader br = new BufferedReader(new FileReader("movies"))) {
                        String line;
                        while ((line = br.readLine()) != null){
                                String columns[] = line.split(",");
                                String movie_id = columns[0].trim();
                                String movie_name = columns[1].trim();
                                movieMap.put(Integer.parseInt(movie_id), movie_name);
                        }
                }
                catch (IOException e){
                        e.printStackTrace();
                }
        }

        if (CacheFiles != null && CacheFiles.length > 1){
                try (BufferedReader br = new BufferedReader(new FileReader("users"))) {
                        String line;
                        while ((line = br.readLine()) != null){
                                String columns[] = line.split(",");
                                String user_id = columns[0].trim();
                                String user_gender = columns[1].trim();
                                userMap.put(Integer.parseInt(user_id), user_gender);
                        }
                }
                catch (IOException e){
                        e.printStackTrace();
                }
        }

}

public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
        String record = value.toString().trim();
        String parts[] = record.split(",");
        String user_id = parts[0].trim();
        String movie_id = parts[1].trim();
        String rating = parts[2].trim();

        String movie_name = movieMap.get(Integer.parseInt(movie_id));
        String user_gender = userMap.get(Integer.parseInt(user_id));

        context.write(new Text(movie_name), new Text(user_gender + " " + rating));
}
}

public static class RatingReducer extends Reducer <Text, Text, Text, Text>
{
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
        ↪  InterruptedException{
                double sum_male_rating = 0.0;
                double sum_female_rating = 0.0;
                int count_male_rating = 0;
                int count_female_rating = 0;
            for (Text record : values) {
                    String parts[] = record.toString().trim().split(" ");
                    String user_gender = parts[0];
                    if (user_gender.equals("F")){
                            sum_female_rating += Double.parseDouble(parts[1]);
                            count_female_rating += 1;
                    }
```

```
                    else if (user_gender.equals("M")){
                            sum_male_rating += Double.parseDouble(parts[1]);
                            count_male_rating += 1;
                    }
            }

            double female_avg = (count_female_rating == 0) ? 0.0 : sum_female_rating /
            ↪  count_female_rating;
            double male_avg = (count_male_rating == 0) ? 0.0 : sum_male_rating /
            ↪  count_male_rating;
            context.write(key, new Text(String.format("Male: %.2f ,Female: %.2f", male_avg,
            ↪  female_avg)));
        }

}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = new Job(conf, "Bai 3");
job.setJarByClass(bai3.class);
job.setMapperClass(RatingMapper.class);
job.setReducerClass(RatingReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
// Setting reducer to zero
//job.setNumReduceTasks(0);


try {

    job.addCacheFile(new URI("hdfs://localhost:8020/mycache/Lab1/movies"));
    job.addCacheFile(new URI("hdfs://localhost:8020/mycache/Lab1/users"));
}
catch (Exception e) {
    System.out.println("File Not Added");
    System.exit(1);
}

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

4. Bài 4

4.1. Minh chứng





4.2. Code

```
import java.io.*;
import java.util.*;
import java.net.URI;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.fs.Path;
import java.net.URI;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;


public class bai4 {

public static class RatingMapper extends Mapper <Object, Text, Text, Text>
{
        private Map<Integer, String> movieMap = new HashMap<>();
        private Map<Integer, String> userMap = new HashMap<>();

public void setup(Context context) throws IOException, InterruptedException
{
        URI[] CacheFiles = context.getCacheFiles();
        System.out.println("Number of cache files: " + (CacheFiles == null ? 0 :
        ↪  CacheFiles.length));

        if (CacheFiles != null && CacheFiles.length > 0){
                try (BufferedReader br = new BufferedReader(new FileReader("movies"))) {
                        String line;
                        while ((line = br.readLine()) != null){
                                String columns[] = line.split(",");
                                String movie_id = columns[0].trim();
                                String movie_name = columns[1].trim();
                                movieMap.put(Integer.parseInt(movie_id), movie_name);
                        }
                }
                catch (IOException e){
                        e.printStackTrace();
                }
        }
```

```
        if (CacheFiles != null && CacheFiles.length > 1){
                try (BufferedReader br = new BufferedReader(new FileReader("users"))) {
                        String line;
                        while ((line = br.readLine()) != null){
                                String columns[] = line.split(",");
                                String user_id = columns[0].trim();
                                String user_age = columns[2].trim();
                                userMap.put(Integer.parseInt(user_id), user_age);
                        }
                }
                catch (IOException e){
                        e.printStackTrace();
                }
        }

}

public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
        String record = value.toString().trim();
        String parts[] = record.split(",");
        String user_id = parts[0].trim();
        String movie_id = parts[1].trim();
        String rating = parts[2].trim();

        String movie_name = movieMap.get(Integer.parseInt(movie_id));
        String user_age = userMap.get(Integer.parseInt(user_id));

        context.write(new Text(movie_name), new Text(user_age + " " + rating));
}
}

public static class RatingReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
    ↪  InterruptedException {
        double sum_0_18 = 0.0, sum_18_35 = 0.0, sum_35_50 = 0.0, sum_50_plus = 0.0;
        int count_0_18 = 0, count_18_35 = 0, count_35_50 = 0, count_50_plus = 0;

        for (Text record : values) {
            String parts[] = record.toString().trim().split(" ");
            int user_age = Integer.parseInt(parts[0]);
            double rating = Double.parseDouble(parts[1]);

            if (user_age <= 18) {
                sum_0_18 += rating;
                count_0_18++;
            } else if (user_age <= 35) {
                sum_18_35 += rating;
                count_18_35++;
            } else if (user_age <= 50) {
                sum_35_50 += rating;
                count_35_50++;
            } else {
                sum_50_plus += rating;
                count_50_plus++;
            }
        }

        String avg_0_18 = (count_0_18 == 0) ? "NA" : String.format("%.2f", sum_0_18 /
        ↪  count_0_18);
        String avg_18_35 = (count_18_35 == 0) ? "NA" : String.format("%.2f", sum_18_35 /
        ↪  count_18_35);
        String avg_35_50 = (count_35_50 == 0) ? "NA" : String.format("%.2f", sum_35_50 /
        ↪  count_35_50);
        String avg_50_plus = (count_50_plus == 0) ? "NA" : String.format("%.2f", sum_50_plus /
        ↪  count_50_plus);

        context.write(key, new Text(String.format("0-18: %s  18-35: %s  35-50: %s  50+: %s",
                avg_0_18, avg_18_35, avg_35_50, avg_50_plus)));
    }
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = new Job(conf, "Bai 4");
job.setJarByClass(bai4.class);
job.setMapperClass(RatingMapper.class);
```

```
job.setReducerClass(RatingReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
// Setting reducer to zero
//job.setNumReduceTasks(0);


try {

    job.addCacheFile(new URI("hdfs://localhost:8020/mycache/Lab1/movies"));
    job.addCacheFile(new URI("hdfs://localhost:8020/mycache/Lab1/users"));
}
catch (Exception e) {
    System.out.println("File Not Added");
    System.exit(1);
}

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```