Big Data Analytics Mini Project
Cinema Data Analysis

Group Members :
Hinal Desai (9)
Mehul Devganiya (11)
Nikita Divay (12)

Abstract :
This mini project aims to collect, display and analyse Cinema data. It focuses on using the IMDB (International Movie DataBase) website to gather information on past statistical data . This data is then analysed and ran using Hadoop Eco-System.
We will be applying Hadoop map-reduce to derive some statistics from IMDB movie data.

Objective:

- To develop essential technical skills such as researching websites, recording data clearly, simple analysis and processing of data.
- To gain experience of handling large amounts of data and presenting it clearly and appropriately.
- To think about and be able to identify outliers in data, how to deal with outliers so they do not skew average results and to think of possible explanations for them.

Introduction :
Big data analytics is the process of examining large data sets containing a variety of data types -- i.e., big data -- to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. The analytical findings can lead to more effective marketing, new revenue opportunities, better customer service, improved operational efficiency, competitive advantages over rival organizations and other business benefits.

In this project we are querying the database loaded in HDFS and retrieveing the records based on the given inputs. We have formed 3 basic questions using which we query the hadoop database. Once the result of the query is given, we get the answer for the required question. Also questions from users can be taken.

Theory :
Big Data
Big data is a term for data sets that are so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying and information privacy. The term often refers simply to the use of predictive analytics or certain other advanced methods to extract value from data, and seldom to a particular size of data set. Accuracy in big data may lead to more confident decision making, and better decisions can result in greater operational efficiency, cost reduction and reduced risk.

Analysis of data sets can find new correlations to "spot business trends, prevent diseases, combat crime and so on." Scientists, business executives, practitioners of medicine, advertising and governments alike regularly meet difficulties with large data sets in areas including Internet search, finance and business informatics. Scientists encounter limitations in e-Science work, including meteorology, genomics, connectomics, complex physics simulations, biology and environmental research.

Data sets are growing rapidly in part because they are increasingly gathered by cheap and numerous

information-sensing mobile devices, aerial (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 exabytes (2.5×1018) of data are created. One question for large enterprises is determining who should own big data initiatives that affect the entire organization.

Big data can be described by the following **characteristics**:

Volume

The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.

Variety

The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.

Velocity

In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

Variability

Inconsistency of the data set can hamper processes to handle and manage it.

Veracity

The quality of captured data can vary greatly, affecting accurate analysis.

he primary goal of big data analytics is to help companies make more informed business decisions by enabling data scientists, predictive modelers and other analytics professionals to analyze large volumes of transaction data, as well as other forms of data that may be untapped by conventional business intelligence (BI) programs. That could include Web server logs and Internet clickstream data, social media content and social network activity reports, text from customer emails and survey responses, mobile-phone call detail records and machine data captured by sensors connected to the Internet of Things. Some people exclusively associate big data with semi-structured and unstructured data of that sort, but consulting firms like Gartner Inc. and Forrester Research Inc. also consider transactions and other structured data to be valid components of big data analytics applications.

Big data can be analyzed with the software tools commonly used as part of advanced analytics disciplines such as predictive analytics, data mining, text analytics and statistical analysis. Mainstream BI software and data visualization tools can also play a role in the analysis process. But the semi-structured and unstructured data may not fit well in traditional data warehouses based on relational databases. Furthermore, data warehouses may not be able to handle the processing demands posed by sets of big data that need to be updated frequently or even continually -- for example, real-time data on the performance of mobile applications or of oil and gas pipelines. As a result, many organizations looking to collect, process and analyze big data have turned to a newer class of technologies that includes Hadoop and related tools such as YARN, MapReduce, Spark, Hive and Pig as well as NoSQL databases. Those technologies form the core of an open source software framework that supports the processing of large and diverse data sets across clustered systems.

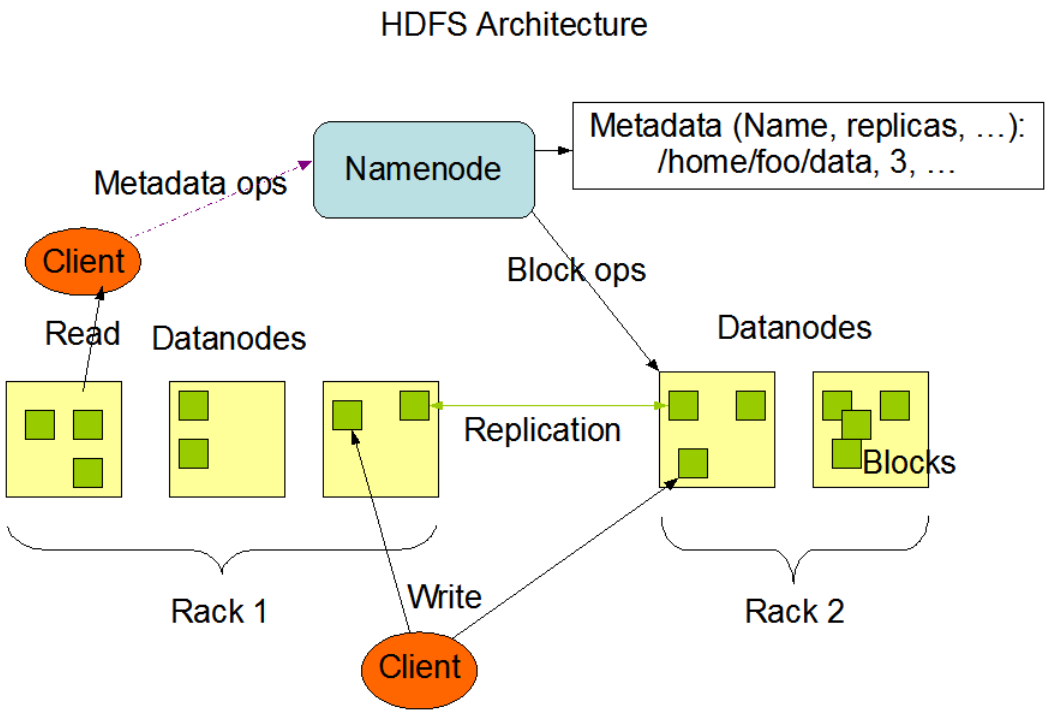
HDFS

HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers. HDFS has demonstrated production scalability of up to 200 PB of storage and a single cluster of 4500 servers, supporting close to a billion files and blocks. When that quantity and quality of enterprise data is available in HDFS, and YARN enables multiple data access applications to process it, Hadoop users can confidently answer questions that eluded previous data platforms.

HDFS is a scalable, fault-tolerant, distributed storage system that works closely with a wide variety of concurrent data access applications, coordinated by YARN. HDFS will “just work” under a variety of physical and systemic circumstances. By distributing storage and computation across many servers, the combined storage resource can grow linearly with demand while remaining economical at every amount of storage.

These specific features ensure that data is stored efficiently in a Hadoop cluster and that it is highly available:

HDFS Architecture



HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system’s clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

`

Goals of HDFS

- **Fault detection and recovery :** Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.

- **Huge datasets :** HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.

- **Hardware at data :** A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

Description

The 3 datasets used here are : movies.dat, users.dat, ratings.dat.

Movies.dat Description

Movie information is in the file "movies.dat" and is in the following format:

MovieID::Title::Genres

- Titles are identical to titles provided by the IMDB (including year of release)
- Genres are pipe-separated and are selected from the following genres:
    * Action
    * Adventure
    * Animation
    * Children's
    * Comedy
    * Crime
    * Documentary
    * Drama
    * Fantasy
    * Film-Noir
    * Horror
    * Musical
    * Mystery
    * Romance
    * Sci-Fi
    * Thriller
    * War
    * Western
- Some MovieIDs do not correspond to a movie due to accidental duplicate entries and/or test entries
- Movies are mostly entered by hand, so errors and inconsistencies may exist

Users.dat Description

User information is in the file "users.dat" and is in the following format:

UserID::Gender::Age::Occupation::Zip-code

All demographic information is provided voluntarily by the users and is

not checked for accuracy. Only users who have provided some demographic information are included in this data set.

- Gender is denoted by a "M" for male and "F" for female
- Age is chosen from the following ranges:
    * 7: "Under 18"
    * 24: "18-24"
    * 31: "25-34"
    * 41: "35-44"
    * 51: "45-55"
    * 56: "55-61"
    * 62: "62+"

- Occupation is chosen from the following choices:
    * 0: "other" or not specified
    * 1: "academic/educator"
    * 2: "artist"
    * 3: "clerical/admin"
    * 4: "college/grad student"
    * 5: "customer service"
    * 6: "doctor/health care"
    * 7: "executive/managerial"
    * 8: "farmer"
    * 9: "homemaker"
    * 10: "K-12 student"
    * 11: "lawyer"
    * 12: "programmer"
    * 13: "retired"
    * 14: "sales/marketing"
    * 15: "scientist"
    * 16: "self-employed"
    * 17: "technician/engineer"
    * 18: "tradesman/craftsman"
    * 19: "unemployed"
    * 20: "writer"

Ratings.dat Description

All ratings are contained in the file "ratings.dat" and are in the following format:

UserID::MovieID::Rating::Timestamp

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- Timestamp is represented in seconds since the epoch as returned by time(2)
- Each user has at least 20 ratings

Questions

1. List all male user id whose age is less than or equal to 7.
   (Using the users.dat file, list all the male userid filtering by age. This demonstrates the use of mapreduce to filter data)

2. Find the count of female and male users in each group.
   (The age distribution is given below :
   *7 : “Under 18”
   *24: “18-24”
   *31: “25-34”
   *41: “35-44”
   *51: “45-54”
   *56: “55-61”
   *62: “62+”
   Use users.dat file.)
3. List all movie title where genre is fantasy.
   (The genre input should be taken fom command line. Use the movies.dat file)

Implementation

1. Apply HADOOP MapReduce to derive some statistics from IMDB Movie Data.
2. Copy Data into Hadoop cluster and use it as input data.
3. One can also use the 'put' or 'copyFromLocal' HDFS shell command to copy those files into HDFS Directory.
4. There are 3 data files : movies.dat, users.dat, ratings.dat.

Steps for Execution

1. All JAR files have same name as class name.
2. Need to include extra jars :
   1. hadoop-common-2.6.0
   2. hadoop-mapreduce-client-core-2.6.0
3. Transfer all required files from local system to VM.
4. Run the following commands :
   1. For Question 1 :
      hdfs dfs -mkdir input1
      hdfs dfs -put users.dat input1
      hadoop jar Question1.jar Question1 input1 output1
   2. For Question 2 :
      hdfs dfs -mkdir input1
      hdfs dfs -put users.dat input1
      hadoop jar Question2.jar Question1 input1 output2
   3. For Question 3 :
      hdfs dfs -mkdir input3
      hdfs dfs -put movies.dat input3
      hadoop jar Question3.jar Question1 input3 output3 Action

To run the jobs use the following command :

hadoop jar name_of_jar_file Classname<input dir><output dir>[<extra input parameter>(may be optional due to question e.g. genre input)]

Source Codes

Question1.java

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
```

```
import org.apache.hadoop.io.Text;
public class Question1 {
  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable>  {
    private final static IntWritable wr = new IntWritable(1);
    private Text area = new Text();
    private String val = null;
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
      String line = value.toString();
      String[] myarray = line.split("::");
                    if(myarray[1].equals("M") && Integer.parseInt(myarray[2]) <= 7)
                    {
        val = myarray[0];
        area.set(val);
        output.collect(area, wr);
                     }
    }
  }
  public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text,
IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }
  public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(Question1.class);
    conf.setJobName("regionCount");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    JobClient.runJob(conf);
  }
}
```

Question2.java

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
```

```
import org.apache.hadoop.io.Text;
public class Question2 {
  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable>  {
    private final static IntWritable wr = new IntWritable(1);
    private Text area = new Text();
    private String val = null;
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
      String line = value.toString();
      String[] myarray = line.split("::");
           if(Integer.parseInt(myarray[2]) <= 18)
           {
     val = "7 "+myarray[1];
     area.set(val);
     output.collect(area, wr);
           }
           else if(Integer.parseInt(myarray[2]) > 18 && Integer.parseInt(myarray[2]) <= 24)
           {
           val = "24 "+myarray[1];
     area.set(val);
     output.collect(area, wr);
           }
           else if(Integer.parseInt(myarray[2]) >= 25 && Integer.parseInt(myarray[2]) <= 34){
            val = "31 "+myarray[1];
     area.set(val);
     output.collect(area, wr);
           }
           else if(Integer.parseInt(myarray[2]) >= 35 && Integer.parseInt(myarray[2]) <= 44){
            val = "41 "+myarray[1];
     area.set(val);
     output.collect(area, wr);
           }
           else if(Integer.parseInt(myarray[2]) >= 45 && Integer.parseInt(myarray[2]) <= 55){
            val = "51 "+myarray[1];
     area.set(val);
     output.collect(area, wr);
           }
           else if(Integer.parseInt(myarray[2]) >= 56 && Integer.parseInt(myarray[2]) <= 61){
            val = "56 "+myarray[1];
     area.set(val);
     output.collect(area, wr);
           }
           else
           {
            val = "62 "+myarray[1];
     area.set(val);
     output.collect(area, wr);
           }
    }
 }
  public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text,
```

```
IntWritable> {
   public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
     int sum = 0;
     while (values.hasNext()) {
       sum += values.next().get();
     }
     output.collect(key, new IntWritable(sum));
   }
  }
  public static void main(String[] args) throws Exception {
   JobConf conf = new JobConf(Question2.class);
   conf.setJobName("regionCount");
   conf.setOutputKeyClass(Text.class);
   conf.setOutputValueClass(IntWritable.class);
   conf.setMapperClass(Map.class);
   conf.setReducerClass(Reduce.class);
   conf.setInputFormat(TextInputFormat.class);
   conf.setOutputFormat(TextOutputFormat.class);
   FileInputFormat.setInputPaths(conf, new Path(args[0]));
   FileOutputFormat.setOutputPath(conf, new Path(args[1]));
   JobClient.runJob(conf);
  }
}
```

Question3.java

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.mapred.*;
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.conf.Configured;
public class Question3 {
  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable>  {
   private final static IntWritable wr = new IntWritable(1);
   private Text area = new Text();
   private static String gen;
   private String val = null;
   @Override
   public void configure(JobConf job) {
      super.configure(job);
      gen= job.get("third_args");
   }
   public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
     String line = value.toString();
     String[] myarray = line.split("::");
     String[]  genre=myarray[2].split("\\|");
          for(int i=0;i<=genre.length-1;i++)
```

```
        {
                if(genre[i].equalsIgnoreCase(gen))
                {
                        val=myarray[1];
                        area.set(val);
                        output.collect(area, wr);
                }
        }
   }
 }
  public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text,
IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }
  public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(Question3.class);
    conf.set("third_args", args[2]);
    conf.setJobName("regionCount");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    JobClient.runJob(conf);
  }
}
```

DataSet
Movies.dat

```
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thriller
11::American President, The (1995)::Comedy|Drama|Romance
12::Dracula: Dead and Loving It (1995)::Comedy|Horror
13::Balto (1995)::Animation|Children's
14::Nixon (1995)::Drama
15::Cutthroat Island (1995)::Action|Adventure|Romance
16::Casino (1995)::Drama|Thriller
17::Sense and Sensibility (1995)::Drama|Romance
18::Four Rooms (1995)::Thriller
19::Ace Ventura: When Nature Calls (1995)::Comedy
20::Money Train (1995)::Action
21::Get Shorty (1995)::Action|Comedy|Drama
22::Copycat (1995)::Crime|Drama|Thriller
23::Assassins (1995)::Thriller
24::Powder (1995)::Drama|Sci-Fi
25::Leaving Las Vegas (1995)::Drama|Romance
26::Othello (1995)::Drama
27::Now and Then (1995)::Drama
28::Persuasion (1995)::Romance
29::City of Lost Children, The (1995)::Adventure|Sci-Fi
30::Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)::Drama
31::Dangerous Minds (1995)::Drama
32::Twelve Monkeys (1995)::Drama|Sci-Fi
33::Wings of Courage (1995)::Adventure|Romance
34::Babe (1995)::Children's|Comedy|Drama
35::Carrington (1995)::Drama|Romance
36::Dead Man Walking (1995)::Drama
37::Across the Sea of Time (1995)::Documentary
```

Users.dat

```
1::M::7::10::48007
2::F::62::16::70002
3::F::31::15::55117
4::F::51::7::02440
5::F::31::20::55445
6::M::56::9::55117
7::F::41::1::06880
8::F::31::12::11443
9::F::31::17::61664
10::M::41::1::95330
11::M::31::1::04003
12::F::31::12::32773
13::F::51::1::93334
14::F::41::0::60116
15::F::31::7::22993
16::M::41::0::20660
17::F::56::1::95330
18::M::24::3::95885
19::F::7::10::48003
20::F::31::14::55113
21::F::24::16::99333
22::F::24::15::53776
23::F::41::0::90009
24::M::31::7::10003
25::F::24::4::01669
26::F::31::7::23112
27::F::31::11::19110
28::M::31::1::14667
29::F::41::7::33447
30::M::41::7::19113
31::F::62::7::06880
32::M::31::0::19335
33::F::51::3::55441
34::M::24::0::02115
35::F::51::1::02442
36::F::31::3::94113
37::M::31::9::66222
```

Ratings.dat

```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
1::595::5::978824268
1::938::4::978301752
1::2398::4::978302281
1::2918::4::978302124
1::1035::5::978301753
1::2791::4::978302188
1::2687::3::978824268
1::2018::4::978301777
1::3105::5::978301713
1::2797::4::978302039
1::2321::3::978302205
1::720::3::978300760
1::1270::5::978300055
1::527::5::978824195
1::2340::3::978300103
1::48::5::978824351
1::1097::4::978301953
1::1721::4::978300055
1::1545::4::978824139
1::745::3::978824268
1::2294::4::978824291
1::3186::4::978300019
1::1566::4::978824330
1::588::4::978824268
1::1907::4::978824330
1::783::4::978824291
1::1836::5::978300172
1::1022::5::978300055
1::2762::4::978302091
1::150::5::978301777
1::1::5::978824268
1::1961::5::978301590
1::1962::4::978301753
1::2692::4::978301570
1::260::4::978300760
```

Results

Question 1

```
[cloudera@quickstart BDA_IMDB_Project]$ hadoop jar Jar/Question1.jar Question1 input1 outputFile1
16/04/12 00:15:57 INFO mapreduce.Job:  map 0% reduce 0%
16/04/12 00:16:07 INFO mapreduce.Job:  map 50% reduce 0%
16/04/12 00:16:08 INFO mapreduce.Job:  map 100% reduce 0%
16/04/12 00:16:14 INFO mapreduce.Job:  map 100% reduce 100%
16/04/12 00:16:14 INFO mapreduce.Job: Job job_1460444418290_0003 completed successfully
[cloudera@quickstart BDA_IMDB_Project]$ hadoop fs -cat /user/cloudera/outputFile1/part-00000
1   1
1045    1
2045    1
210 1
2133    1
2137    1
2155    1
5844    1
5989    1
6006    1
606 1
629 1
634 1
743 1
75  1
86  1
888 1
99  1
```

Question2

```
[cloudera@quickstart BDA_IMDB_Project]$ hadoop jar Jar/Question2.jar Question2 input2 outputFile2

[cloudera@quickstart BDA_IMDB_Project]$ hadoop fs -cat /user/cloudera/outputFile2/part-00000
24 F    805
24 M    298
31 F    1538
31 M    558
41 F    855
41 M    338
51 F    361
51 M    189
56 F    350
56 M    146
62 F    278
62 M    102
7 F 144
7 M 78
```

Question3

```
cloudera@quickstart BDA_IMDB_Project]$ hadoop jar Jar/Question3.jar Question3 input3 outputFile3 Action

16/04/12 00:14:37 INFO mapreduce.Job:  map 0% reduce 0%
16/04/12 00:14:46 INFO mapreduce.Job:  map 50% reduce 0%
16/04/12 00:14:47 INFO mapreduce.Job:  map 100% reduce 0%
16/04/12 00:14:53 INFO mapreduce.Job:  map 100% reduce 100%
16/04/12 00:14:53 INFO mapreduce.Job: Job job_1460444418290_0002 completed successfully

[cloudera@quickstart BDA_IMDB_Project]$ hadoop fs -cat /user/cloudera/outputFile3/part-00000

13th Warrior, The (1999)    1
3 Ninjas: High Noon On Mega Mountain (1998) 1
52 Pick-Up (1986)   1
7th Voyage of Sinbad, The (1958)    1
Abyss, The (1989)   1
Aces: Iron Eagle III (1992) 1
Action Jackson (1988)   1
Adrenalin: Fear the Rush (1996) 1
Adventures of Robin Hood, The (1938)    1
Wisdom (1986)   1
World Is Not Enough, The (1999) 1
Wrongfully Accused (1998)   1
X-Men (2000)    1
Young Guns (1988)   1
Young Guns II (1990)    1
Young Sherlock Holmes (1985)    1
Zero Kelvin (Kj◆rlighetens kj◆tere) (1995)  1
eXistenZ (1999) 1
```

Conclusion

Hence with this project we have understood the need of Big Data and also understood the basic HDFS architecture and its working. We have also learnt the use of mapreduce for querying the database and obtaining the result. Here we have loaded our data set into hadoop database and queried it to get desired output using mapreduce code.