Analysis of Earthquake Occurrences Using Hadoop

Vikramaditya Pagadala, Ananthi Selvamani, Kislay Jha
Department of Computer Science
University of Bridgeport
Connecticut, United States of America
vpagadal@my.bridgeport.edu
aselvama@my.bridgeport.edu
kjha@my.bridgeport.edu

Abstract: *- Hadoop has been an effective framework to store and process huge volume of data in a distributed environment. So on the earthquake data set collected from the site U.S Geological Survey for the year 1940 to 2016 this framework is used and reliable outputs are generated from it. The Earthquake data includes various parameters in it so a Map Reduce programming model available in the Hadoop is used on this data set which gives the total number of earthquake occurrences on a given criteria. The output from the Map Reduce program is processed with Hive and Spark techniques from which we can finally get the total number of earthquake occurrences in a specific area.*

1. Introduction

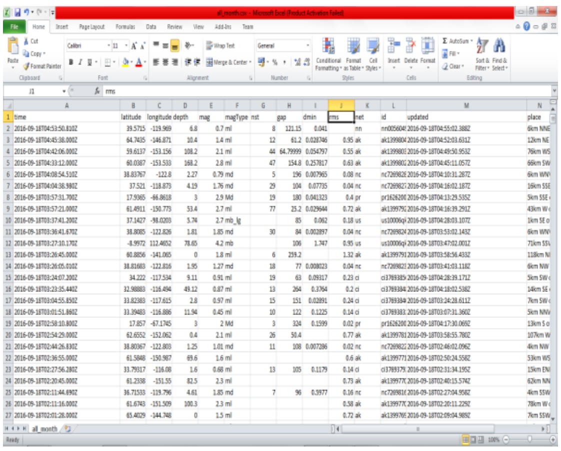
The U.S Geological Survey site [2] records all the earthquake occurrences which has been occurred for a very long time. These records are a source of valuable information because by analyzing these data we can come to a prediction of an earthquake event before it occurs and what are the necessary precautions to be taken in an area when an earthquake occurs. But the real problem lies with the size of this data. This earthquake data contains various characteristics information of which the most important characteristics are explained below:

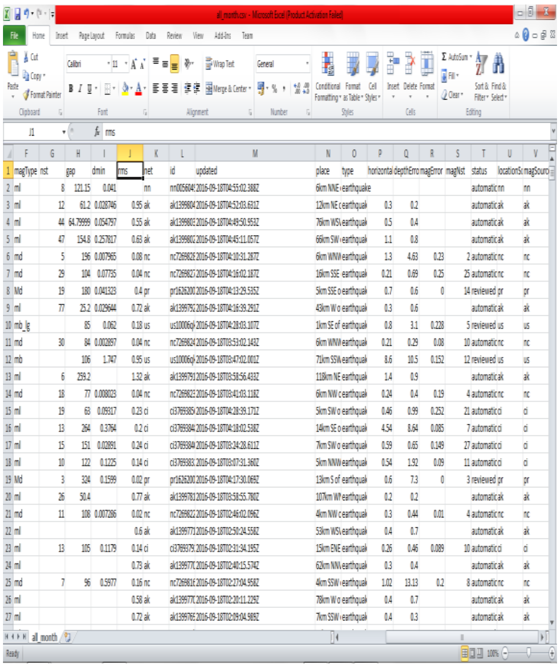
- Time – Time when the event has occurred reported in milliseconds.
- Latitude-Longitude - Typical values [-90.0, 90.0] ,[-180.0 180.0]
- Depth – Depth of the event in kilometers.
- Mag – Magnitude of the event –typical values [-1.0, 10.0].
- MagType – Preferred magnitude of an event. Typical values are md, mi , ms, mw, me, ml, mb, mlg.
- Gap – The largest azimuthal gap between azimuthally adjacent stations. The smaller the value the reliable the calculated earthquake position.
- Place- The textual description of the named geographic region near the event.
- Type –Type of Seismic event. Typical values – Earthquake, quarry etc.,
-

To process these information will be a tedious task as there are various readings recording for a single day. So to process this information in a easier way the hadoop framework is used which uploads these data in to HDFS through which we can process the data available in various clusters and obtain reliable result from it.

II.Dataset

The Data for this project is taken from the site USGS where the data can be queried for a year and the data will be retrieved in the format of csv. All these data files are combined into one excel file which contains the records from the year 1940 to 2016 January. A screenshot of the data set is shown below:





III. Data Characteristics

The following are the data characteristics of the earthquake dataset:

Time :
Data Type - Long Integer
Description - Time when the event occurred. Times are reported in *milliseconds* since the epoch (1970-01-01T00:00:00.000Z), and do not include leap seconds. In certain output formats, the date is formatted for readability.

Latitude :
Data Type – Decimal
Typical Values[-90.0, 90.0]
Description - Decimal degrees latitude. Negative values for southern latitudes

Longitude :
Data Type – Decimal
Typical Values - [-180.0, 180.0]
Description - Decimal degrees longitude. Negative values for western longitudes.

Depth :
Data Type – Decimal
Typical Values - [0, 1000]
Description - Depth of the event in kilometers.

Mag:
Data Type – Decimal
Typical Values - [-1.0, 10.0]
Description - The magnitude for the event. Learn more about magnitudes

MagType:
Data Type – String
Typical Values - “Md”, “Ml”, “Ms”, “Mw”, “Me”, “Mi”, “Mb”, “MLg”
Description - The method or algorithm used to calculate the preferred magnitude for the event.Learn more about magnitude types.

Nst:
Data Type – Integer
Description - The total number of seismic stations used to determine earthquake location.

Gap:
Data Type – Decimal
Typical Values - [0.0, 180.0]
Description - The largest azimuthal gap between azimuthally adjacent stations (in degrees). In general, the smaller this number, the more reliable is the calculated horizontal position of the earthquake.

Dmin:
Data Type – Decimal
Typical Values - [0.4, 7.1]
Description - Horizontal distance from the epicenter to the nearest station (in degrees). 1 degree is approximately 111.2 kilometers. In general, the smaller this number, the more reliable is the calculated depth of the earthquake.

Rms:
Data Type – Decimal
Typical Values - [0.13,1.39]
Description - The root-mean-square (RMS) travel time residual, in sec, using all weights. This parameter provides a measure of the fit of the observed arrival times to the predicted arrival times for this location. Smaller numbers reflect a better fit of the data. The value is dependent on the accuracy of the velocity model used to compute the earthquake location, the quality weights assigned to the arrival time data, and the procedure used to locate the earthquake.

Net:
Data Type – String
Typical Value - sak, at, ci, hv, ld, mb, nc, nm, nn, pr, pt, se, us, uu, uw
Description - The ID of a data contributor. Identifies the network considered to be the preferred source of information for this event.

Place:
Data Type – String
Description - Textual description of named geographic region near to the event. This may be a city name, or a Flinn-Engdahl Region name.

Type:
Data Type – String

Typical Values - “earthquake”, “quarry”
Description - Type of seismic event.

HorizontalError:
Data Type – Decimal
Typical Values - [0, 100]
Description - Uncertainty of reported location of the event in kilometers.

DepthError:
Data Type – Decimal
Typical Values - [0, 100]
Description - Uncertainty of reported depth of the event in kilometers.

MagError:
Data Type – Decimal
Typical Values - [0, 100]
Description - Uncertainty of reported magnitude of the event.

MagNst:
Data Type – Integer
Description - The total number of seismic stations used to calculate the magnitude for this earthquake.

Status:
Data Type – String
Typical Values - “automatic”, “reviewed”, “deleted”
Description - Indicates whether the event has been reviewed by a human.

IV. MapReduce

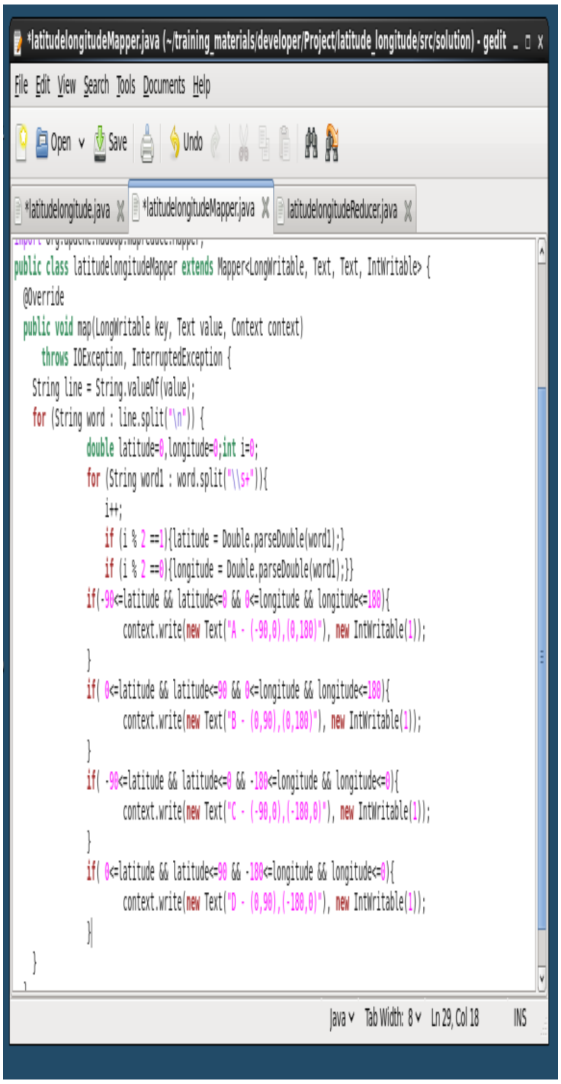
The research paper is about Map Reduce design[1] and its implementation in processing a large amount of datasets. In this model there are two phase (1) Map phase which creates an intermediate value based on the key/value pair and (2) phase is Reduce phase which uses these intermediate key and merges all its values. The map reduce programs are parallelized and to there are executed on a large group of machines called clusters.

The Map Reduce program is highly scalable and can be executed on a cluster of machines which can handle huge volume of data. For instance when huge amount of data has to be processed on a cluster of machines in case of any failure of the machines in the cluster then the data processing has to be handled properly. This involves some tedious task so to overcome this we have a abstracted technique which is Map Reduce which creates some intermediate key/values in the map phase and in the reduce phase these intermediate key values are merged to get a result. So in this research paper the MapReduce technique is described in detail which involves the implementation of it in large clusters, the working of its programming model, performance measurements of this technique in various workloads.

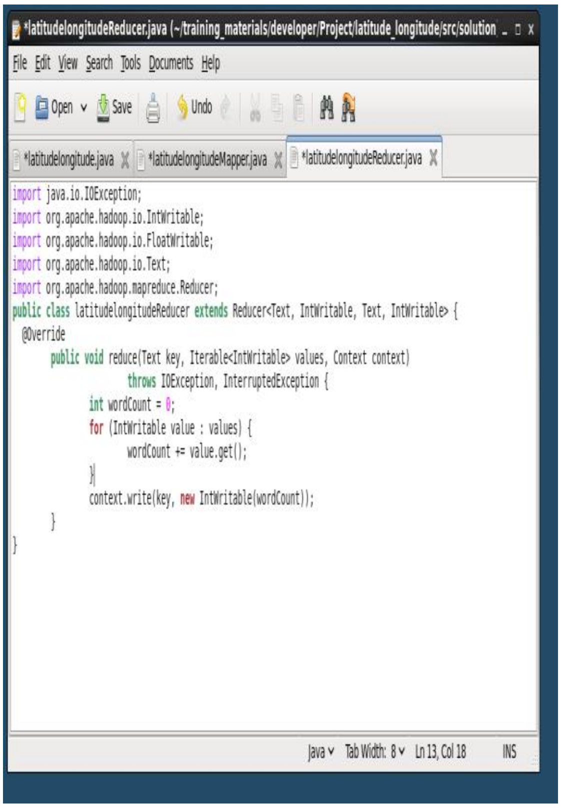
The map reduce function first splits the input file into M pieces of size 64mb. The library then starts copies of these splits in many machines. Out of these copies one of them acts as a master which assigns task to other worker machines. The worker process this split and generates key value pairs which are written in the memory. The partitioning function plays a major role in writing this intermediate result to the memory and notify the master regarding this. The master in turn notifies the reducer worker about this location and the reducer uses sort functionality to process this intermediate result and creates a final output. The output result is then appended to the output file. Once the map reduce is done the master notifies the user program about the output file.

The Mapper, Reducer, Driver code for the latitude longitude word count program is shown below:

Mapper Code:

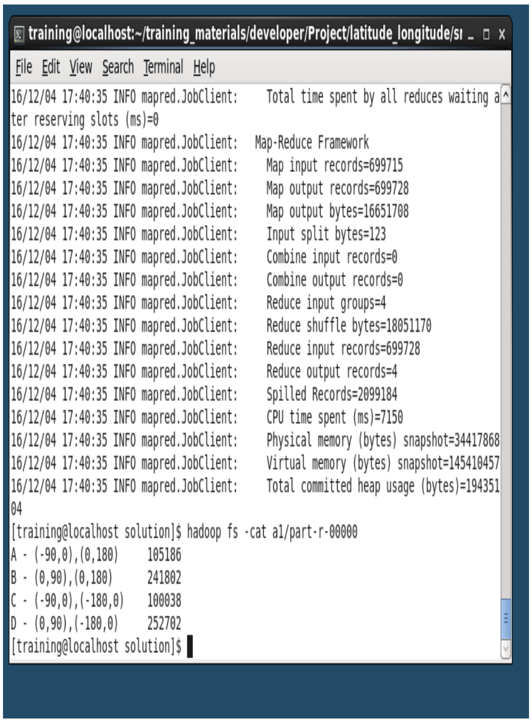


Reducer Code:



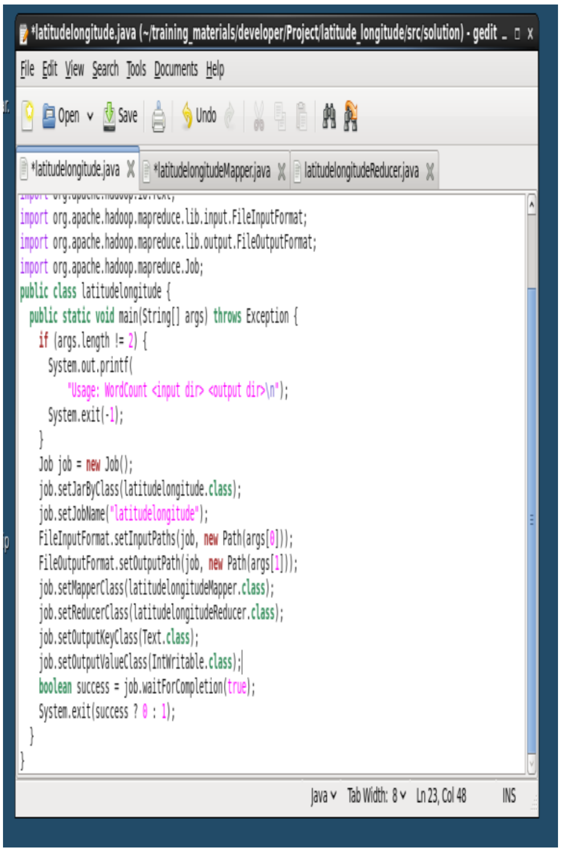
```
*latitudelongitudeReducer.java (~/training_materials/developer/Project/latitude_longitude/src/solution
File Edit View Search Tools Documents Help
*latitudelongitude.java  *latitudelongitudeMapper.java  *latitudelongitudeReducer.java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class latitudelongitudeReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
 @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
        int wordCount = 0;
        for (IntWritable value : values) {
            wordCount += value.get();
        }
        context.write(key, new IntWritable(wordCount));
    }
}
Java  Tab Width: 8  Ln 13, Col 18  INS
```

The output when the map reduce program runs will be as follows:



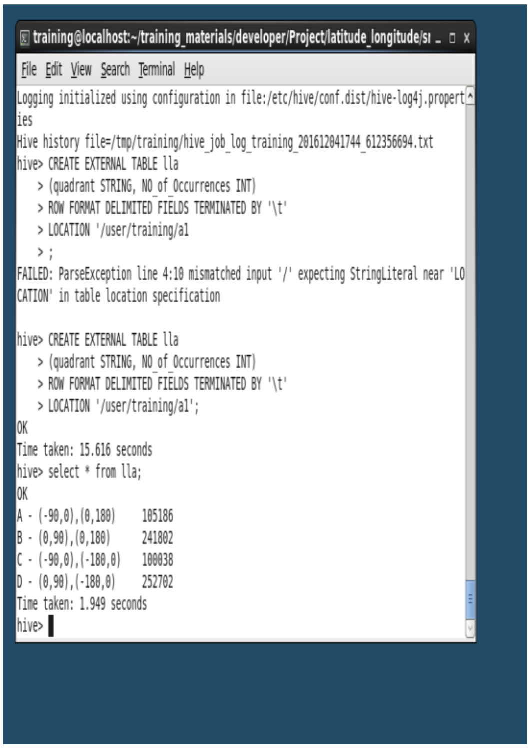
```
training@localhost:~/training_materials/developer/Project/latitude_longitude/s
File Edit View Search Terminal Help
16/12/04 17:40:35 INFO mapred.JobClient:     Total time spent by all reduces waiting a
fter reserving slots (ms)=0
16/12/04 17:40:35 INFO mapred.JobClient:   Map-Reduce Framework
16/12/04 17:40:35 INFO mapred.JobClient:     Map input records=699715
16/12/04 17:40:35 INFO mapred.JobClient:     Map output records=699728
16/12/04 17:40:35 INFO mapred.JobClient:     Map output bytes=16651708
16/12/04 17:40:35 INFO mapred.JobClient:     Input split bytes=123
16/12/04 17:40:35 INFO mapred.JobClient:     Combine input records=0
16/12/04 17:40:35 INFO mapred.JobClient:     Combine output records=0
16/12/04 17:40:35 INFO mapred.JobClient:     Reduce input groups=4
16/12/04 17:40:35 INFO mapred.JobClient:     Reduce shuffle bytes=18051170
16/12/04 17:40:35 INFO mapred.JobClient:     Reduce input records=699728
16/12/04 17:40:35 INFO mapred.JobClient:     Reduce output records=4
16/12/04 17:40:35 INFO mapred.JobClient:     Spilled Records=2099184
16/12/04 17:40:35 INFO mapred.JobClient:     CPU time spent (ms)=7150
16/12/04 17:40:35 INFO mapred.JobClient:     Physical memory (bytes) snapshot=34417868
16/12/04 17:40:35 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=145410457
16/12/04 17:40:35 INFO mapred.JobClient:     Total committed heap usage (bytes)=194351
04
[training@localhost solution]$ hadoop fs -cat a1/part-r-00000
A - (-90,0),(0,180)     105186
B - (0,90),(0,180)      241802
C - (-90,0),(-180,0)    100038
D - (0,90),(-180,0)     252702
[training@localhost solution]$
```

Driver Code:



```
*latitudelongitude.java (~/training_materials/developer/Project/latitude_longitude/src/solution) - gedit
File Edit View Search Tools Documents Help
*latitudelongitude.java  *latitudelongitudeMapper.java  latitudelongitudeReducer.java
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
public class latitudelongitude {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.out.printf(
        "Usage: WordCount <input dir> <output dir>\n");
      System.exit(-1);
    }
    Job job = new Job();
    job.setJarByClass(latitudelongitude.class);
    job.setJobName("latitudelongitude");
    FileInputFormat.setInputPaths(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(latitudelongitudeMapper.class);
    job.setReducerClass(latitudelongitudeReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    boolean success = job.waitForCompletion(true);
    System.exit(success ? 0 : 1);
  }
}
Java  Tab Width: 8  Ln 23, Col 48  INS
```

The Hive implementation of the same code is shown below:

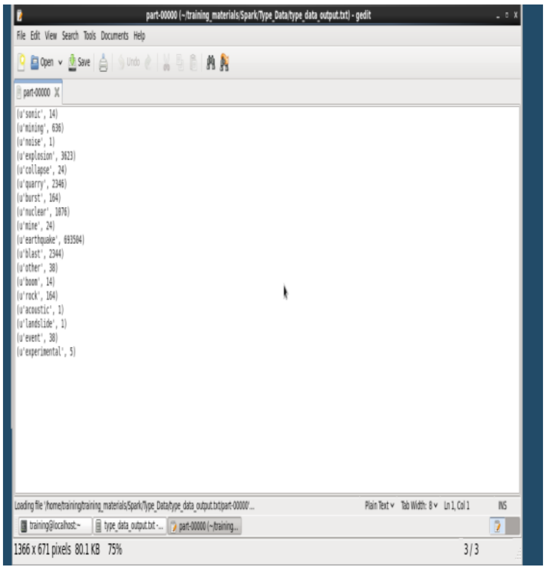


```
training@localhost:~/training_materials/developer/Project/latitude_longitude/s
File Edit View Search Terminal Help
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.propert
ies
Hive history file=/tmp/training/hive_job_log_training_201612041744_612356694.txt
hive> CREATE EXTERNAL TABLE lla
    > (quadrant STRING, NO_of_Occurrences INT)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > LOCATION '/user/training/a1
    > ;
FAILED: ParseException line 4:10 mismatched input '/' expecting StringLiteral near 'LO
CATION' in table location specification

hive> CREATE EXTERNAL TABLE lla
    > (quadrant STRING, NO_of_Occurrences INT)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > LOCATION '/user/training/a1';
OK
Time taken: 15.616 seconds
hive> select * from lla;
OK
A - (-90,0),(0,180)     105186
B - (0,90),(0,180)      241802
C - (-90,0),(-180,0)    100038
D - (0,90),(-180,0)     252702
Time taken: 1.949 seconds
hive>
```

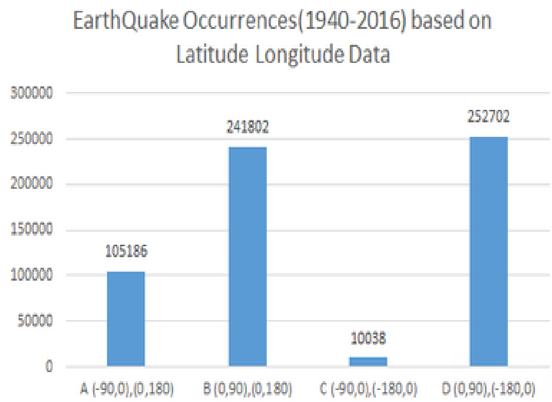
SPARK:

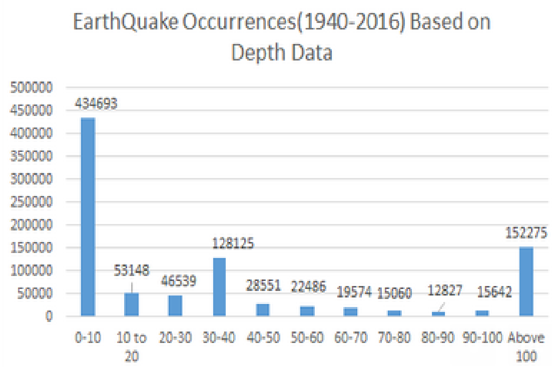
The same work can be run in a SPARK framework which is an application program interface based on a data structure called as resilient distributed dataset(RDD). A spark object has to be created first and in this the map reduce code has be written which will provide the result as key value pairs

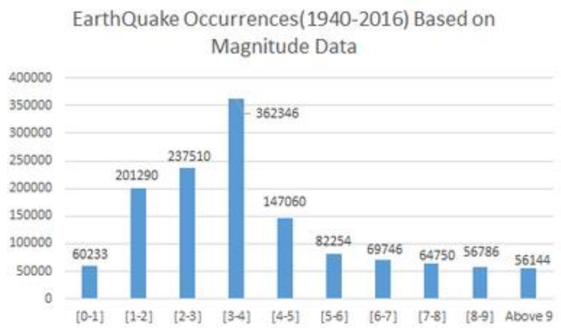


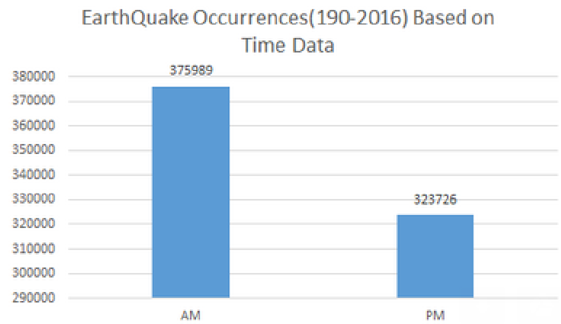
V. Results and Evaluation

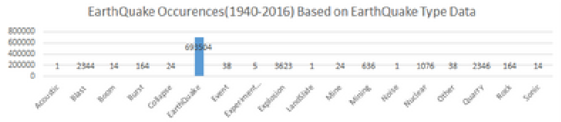
The above mentioned Map Reduce Program can be run on various characteristics of the Earthquake dataset and the result retrieved from it is presented below:

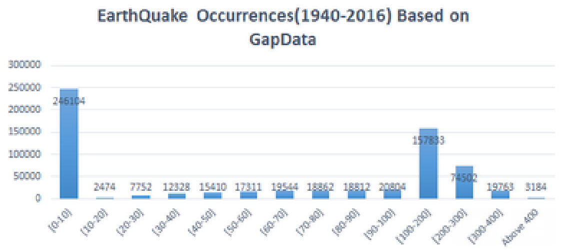












VI. Conclusion:

As observed from the output of the above map reduce code we conclude that hadoop framework can be used to store and process data of huge volume. The output from the map reduce code will be key value pairs which are used to find some pattern in the output and with this pattern we predict the occurrences of earthquake in a given place.

References:

1. *Jeffrey Dean and Sanjay Ghemawat* MapReduce : Simplified Data Processing on Large Clusters

2. *U.S Geological Survey https://www.usgs.gov/*