ABSTRACT:

An earthquake is shaking of the surface of the Earth, which caused as the result of movable plate boundary interactions. The sudden release of energy called seismic waves that destroys so many buildings and kills thousands of people.

Earthquakes are measured using remarks from seismometers with Richter magnitude scale. Ground rupture, Landslides, Soil liquefaction and Tsunami are the main effects created by earthquakes. Today's earthquake warning systems used to provide regional notification of an earthquake in progress. Many methods have been already developed for predicting the time and place in which earthquakes will occur, but it did not predicted using big data analytics.

Our Project addresses the novel methodology to identify next earthquake happening from tons of international geological survey data using Map Reduce model. In addition, Map and Reduce function used to locate highest shaky place, location nearer to fault line, current location shakes per minute. Other than above mentioned features separate Map and Reduce function implemented to analyze sheer number of earthquakes per day. Final result shows which location suffered from maximum number of shakes and priority of earthquake occurrence location.

EARTHQUAKE ANALYTICS:

A. Objective:

Earthquake analytics explored in this project attempt to predict the location of a next earthquake based on U.S. Geological Survey Data.

B. Implementation:

In this proposed system next earthquake prediction is performed with following steps.
Initial step to start with is Data Collection:(My part of the work)
Collected data from various sources like kaggle ,U.S Geology Survey Centre ,USGS Science for a changing world are examples of few such sources.

Step 1: Data parsing

Step 2: Date format conversion
Step 3: Define Map function
Step 4: Define Reduce function
Step 5: Define Hadoop Job

C. Data parsing: (My part of the work)

First step towards earthquake prediction is combining all the found data and parsing the earthquake CSV file. The starting line of information rich CSV file is called header. In this file each line is series of data values separated by commas. The proposed analytics work is primarily interested in three pieces of data: date, location and magnitude of each earthquake. To parse the above mentioned parameter needs open source library named opencsv, which is used to parsing CSV files. The parser opencsv makes working with comma separated values very simple. This parser simply returns an array of strings, so it is possible to obtain positional values. The following code with JUnit test is used to parse earthquake CSV file.

```
public class CSVParsing {
    private final String L = "ci,14897012,2,\"Monday, " +
                             "December 13, 2010 " +
                             "14:10:32 UTC\",33.0290,-115." +
                             "5388,1.9,15.70,41," +
                             "\"Southern California\"";

    public void testReadingOneLine() throws Exception
    {
        String[ ] lines = new CSVParser().parseLine(L);
        assertEquals("should be Monday, December 13, 2010 " +
                     "14:10:32 UTC","" +
                     "" + "December 13, 2010 14:10:32 UTC", lines[3]);
        assertEquals("should be Southern California","" +
                     "Southern California", lines[9]);
        assertEquals("should be 1.9", "1.9", lines[6]);
    }
```

D. Date format conversion:

In this proposed system earthquake prediction is performed using MapReduce model. While working with this model, the user defined function map is used to pick some values along with some key. The map function returns collection of key value pairs as

output. In this system map function to determine earthquake considers date and counter as key-value pair. Date format conversion function is mainly used for standardization.
In the earthquake CSV file date will be the key and counter will be the value. The user defined function Reduce sum up the counters value that provides number of occurrence of date in a target CSV earthquake data. The date string in the CSV file is converted using simple date format java object. The simple date format should be dd-mm-yyyy.
The following code converts the date string into specific date format.

```
public void DateformatConversion() throws Exception
{
String dt = "Monday, December 13, 2010 14:10:32 UTC";
SimpleDateFormat formatter = new SimpleDateFormat(
                  "EEEEE, MMMMM dd, yyyy HH:mm:ss Z");
Date d = formatter.parse(dt);
formatter.applyPattern("dd-MM-yyyy");
String dts = formatter.format(d);
assertEquals("should be 13-12-2010", "13-12-2010", dts);
}
```

E. Live streaming Data:(My part of the work)

Along with the static data available Live Streaming data has been collected from:
https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_day.geojson
Here the data extracted will be in JSON format use JSON to text converter to convert the data into text format and can further be stored in a .csv file. This data keeps getting refreshed everytime we reload the browser , most recent data will be appended to the existing data on every refresh.
Every minute shake will be covered in this data.Live data extracted in every attempt is given to json formatter as shown:

The text format obtained is saved as CSV file and given to Data Formatting phase.
My Work involves defining every function theoretically on how a function should behave .Here , functions are map function,reduce function and hadoop job.

F. Define map function:(My part of the work)

The map function present in the java file named EarthquakesPerDayMapper.java is used to identify the number of earthquake occurs per day. The map function of hadoop should be extending with Mapper class of hadoop. Map function also uses generics to explicitly specify the key-value present in CSV file. In this proposed analysis work system main class EarthquakePerDayMapper is an extension to Mapper object. This class contains output key as a Text object type and its value as an IntWritable type, which is hadoop specific class similar to integer. LongWritable and Text used for byte count and line of text. Because of type clause in EarthquakesPerDayMapper, various parameters coming into map function are set along with the output of this function inside context.write clause. The following code shows EarthquakesPerDay map function.

```java
public class EarthquakesPerDayMapper extends
Mapper<LongWritable, Text, Text, IntWritable>
{
protected void map(LongWritable key, Text value,
        Context context) throws IOException,
        InterruptedException {
if (key.get() > 0) {
try {
CSVParser parser = new CSVParser();
String[ ] lines = parser.parseLine(value.toString());
SimpleDateFormat formatter =  new SimpleDateFormat
                             ("EEEEE, " +"MMMMM dd, " +
                              "yyyy HH:mm:ss Z");
Date dt = formatter.parse(lines[3]);
formatter.applyPattern("dd-MM-yyyy");
String dts = formatter.format(dt);
context.write(new Text(dts), new IntWritable(1));
} catch (ParseException e) {}
}
}
}
```

The above mentioned map function is Apache hadoop invokes the EarthquakesPerDayMapper for each line of text it finds in an input CSV file. To avoid the header of CSV file, Mapper class checks if the byte count is not zero. Date conversion can be performed using SimpleDateFormat and set it as outgoing key. The value count is set to 1 in the same line. The key-value pair generated for all possible dates. The logical view of output of the map function and input for reduce function is given below:

"13-12-2010":[1,1,1,1,1,1,1,1]
"14-12-2010" :[1,1,1,1,1,1]
"15-12-2010" :[1,1,1,1,1,1,1,1,1]

Note that the line context.write statement with Text and IntWritable parameter in map function creates above logical collection. The clause Context is a hadoop data structure that contains various collection of information. This context output is passed to the reduce function as input.

G. Define reduce function:(My part of the work)

Similar to EarthquakesPerDayMapper.java Mapper class Reducer class present in EarthquakesPerDayReducer.java extends hadoop Reducer object. This class contains output key as a Text object type and its value as an IntWritable type, which is hadoop specific class. The input and output parameter object types are same. The output parameter date represented as Text object and count represented as IntWritable. In the proposed EarthquakePerDayReducer implementation, the incoming value must be collection of values .In this case collection has number of one's Here one represent earthquake occurrence. The reducer function simply counts number of occurrences. Then Reducer creates new key value pair that represents date and sum. The clause context in the reduce function returns key-value pair that contains date as the key and sum as the value. The following code shows EarthquakesPerDayReducer class.

```
public class EarthquakesPerDateReducer extends
Reducer<Text, IntWritable, Text, IntWritable> {

protected void reduce(Text key, Iterable<IntWritable>
                      values, Context context)
throws IOException, InterruptedException {
int count = 0;
for (IntWritable value : values) {
count++;
}
context.write(key, new IntWritable(count));
}
}
```

H. Define hadoop job:(My part of the work)

Hadoop job link coded map and reduce implementation. Defining a Hadoop job is brutally simple. It needs inputs, outputs, Mapper implementation, Reducer implementation and output types. In this system output types are same as output types used in reduce implementation. In EarthquakesPerDayJob class tied everything together with a main function that takes directory of earthquake CSV file and result as parameters. The following code shows hadoop job function for Earthquakes per day analytics.

```
public class EarthquakesPerDayJob {
public static void main(String[ ] args) throws Throwable {
Job job = new Job();
job.setJarByClass(EarthquakesPerDayJob.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.setMapperClass(EarthquakesPerDayMapper.class);
job.setReducerClass(EarthquakesPerDayReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

I. Define map function for location:(My part of the work)

The second Mapper code present in the file named EarthquakePerLocationMapper.java is used to identify the number of earthquake occurs by location. Map function also uses generics to explicitly specify the key value pair. Similar to previous code the input for map function is byte count and line of text present in CSV file. In this proposed work main class EarthquakePerLocationMapper extends to Mapper object. Here date retrieval and conversion part of previous analysis code is eliminated. The map function uses location field from CSV array.

```
public class EarthquakePerLocationMapper extends
Mapper<LongWritable, Text, Text, IntWritable> {
protected void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
if (key.get() > 0) {
String[ ] lines = new CSVParser().parseLine(value.toString());
context.write(new Text(lines[9]), new IntWritable(1));
}
}
}
```

J. Define reduce for next earthquake:(My part of the work)

Similar to new Mapper class proposed second Reducer class in EarthquakePerLocationReducer class used to identify the next earthquake possible location based on number of earthquake occurs in particular location. This analysis is done using EarthquakePerLocationMapper output.The reducer function limits the result to any location with 15 or more earthquake in a week.

```java
public class EarthquakePerLocationReducer extends
Reducer<Text, IntWritable, Text, IntWritable> {

protected void reduce(Text key, Iterable<IntWritable>
                  values, Context context)
throws IOException, InterruptedException {
int count = 0;
for (IntWritable value : values) {
count++;
}
if (count >= 10)
{
    context.write(key, new IntWritable(count));
}
}
}
```

Knowledge and Data Visualization part of the Project Diagram:

K. Output - earthquakes by date:

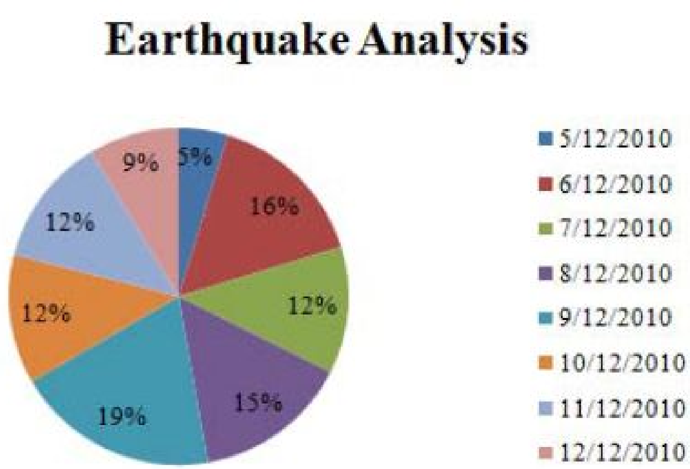
```
hduser@ubuntu: ~
hduser@ubuntu:~$ hadoop fs -cat /user/hduser/earthqua
e/out/part-r-00000
05-12-2010      43
06-12-2010      143
07-12-2010      112
08-12-2010      136
09-12-2010      178
10-12-2010      114
11-12-2010      114
12-12-2010      79
```

L. Output - earthquakes by location:

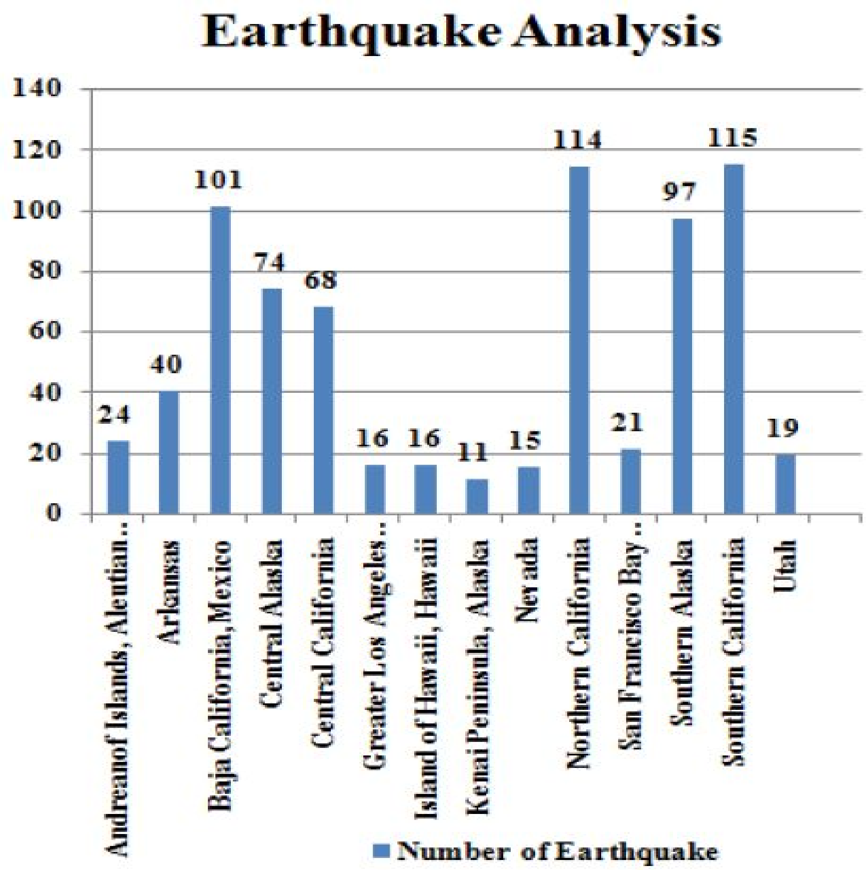
```
hduser@ubuntu: ~
hduser@ubuntu:~$ hadoop fs -cat /user/hduser/earthqua
e/out/part-r-00000
Andreanof Islands, Aleutian Islands, Alaska     24
Arkansas        40
Baja California, Mexico 101
Central Alaska  74
Central California      68
Greater Los Angeles area, California    16
Island of Hawaii, Hawaii        16
Kenai Peninsula, Alaska 11
Nevada  15
Northern California     114
San Francisco Bay area, California      21
Southern Alaska 97
Southern California     115
Utah    19
```

RESULT AND DISCUSSION:

This section describes about earthquake analysis, next earthquake discovery based on proposed map and reduce function result. By using second map and reduce functions highest shaky place, location nearer to fault line and current location shakes per minute are predicted.



Earthquake analysis with EarthquakesPerDayMapper and Reducer output.



Earthquake analysis using EarthquakePerLocationMapper and Reducer output.

The chart in figure is generated using the result received from proposed EarthquakePerLocationMapper and Reducer class. In the above chart X-axis contains locations and Y-axis contains number of earthquakes. With the help of generated graph highest shaky place is, location nearer to fault line and current location shakes per minute are identified easily. Based on sample USGS data set (2010) Alaska from Mexico is a shaky place. Highest shaky place and next possible earthquake location is Southern California. Arkansas apparently sits near a fault line. Finally, if anyone lives in Northern or Southern California then the ground shakes every 13 minutes.