Analysis of USGS Earthquake Data

NANDITA SATHE[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: nsathe@iu.edu*

project-001, April 25, 2017

Big Data Analytics tools allow us to analyze the huge volumes of geo-spatial data. Data of earthquakes that take place globally is a major part of crucial geo-spatial data. US Geological Survey's (USGS) Earthquake Hazards Program monitor and report earthquakes, assess earthquake impacts and hazards, and research the causes and effects of earthquakes [1]. Machine learning algorithms are used for advanced data analysis and earthquakes prediction.

Keywords: I524, geospatial, MongoDB, Plotly, K-means clustering, DBSCAN, Python, pymongo, USGS, Ansible, Cherrypy

https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IO-3017/report/report.pdf

1. INTRODUCTION

USGS collects volumes of geospatial data pertaining to earthquakes and makes it available for analysis. The project obtains a chunk of the data using GeoJSON web service provided by USGS. The data is saved locally in MongoDB database. Data is analysed using machine learning algorithms. The output is plotted (rendered) on web browser. A light weight web server is used to respond to the web requests. Project is capable of running in cloud environment. Deployment is automated using Ansible.

2. TECHNOLOGIES USED

Technologies used for development and deployment of this project are listed below.

1. **Cloudmesh** - Cloudmesh provides Cloud Testbeds as a Service (CTaaS). It is a platform where one can manage all cloud accounts and local files enabling one to copy them between services. Project uses cloudmesh to connect to various cloud environments.

2. **Ansible** - Ansible is an IT automation tool that automates cloud provisioning, configuration management, and application deployment. Once Ansible gets installed on a control node, which is an agentless architecture, it connects to a managed node through the default OpenSSH connection type. Ptoject uses ansible for one-click deployment.

3. **Python** - Python is an object oriented, light weight programming language. Python is primary programming language used in this project.

4. **Mongo-DB** - MongoDB is an unstructured (NOSQL) database, which uses document-oriented data model. It stores data in flexible JSON-like documents. The project uses Mongo-DB to store GeoJSON data of earthquakes locally.

5. **Plotly** - Plotly is data analytics and visualization tool. One can create interactive graphs using plotly. It provides graphing libraries for Python. The plotly is used in the project as a visualization tool.

6. **Scikit-learn** - Scikit-learn provides tools for data analysis and data mining. Scikit-learn provides a wide range of learning algorithms. Scikits are the names given to the modules for SciPy, a fundamental library for scientific computing. As these modules provide different learning algorithms, the library is named as sciki-learn. In this project data classification is done using scikit learn.

7. Cherrypy - CherryPy is an object-oriented web application framework. It is designed for rapid development of web applications by wrapping the HTTP protocol. It is WSGI (Web Server Gateway Interface) thread-pooled web server. Cherrypy is used as a web server in the application.

3. IMPLEMENTATION

3.1. USGS Web Service and Data Set

USGS earthquake data is fetched using the web service and passing relevant parameters to it. Out of the voluminous world-wide data of decades this project uses data of earthquakes appeared in North and South America for duration of 2 years from 2015-1-1 till 2017-1-1, having magnitude over 4. To select North and South America region data, its Latitude and Longitude are taken from Search Earth Catalog tool provided by USGS. [2]. Web service returns data in JSON format.

3.2. Data Processing and Visualization

Severity of earthquakes. Severity of earthquakes is analysed by their magnitude and depth. Data is classified into clusters using K-Means clustering algorithm (Section 4.1). Clusters are plotted on a interactive scatter plot.

Region affected the most by earthquakes. Analysing Latitude (lat)-Longitude (lon) of epicenter of the earthquake shows the regions where earthquakes are frequent. Lat lon data is clustered using DBSCAN algorithm (Section 4.2). Clusters are plotted on a geo-scatter plot on a world map.

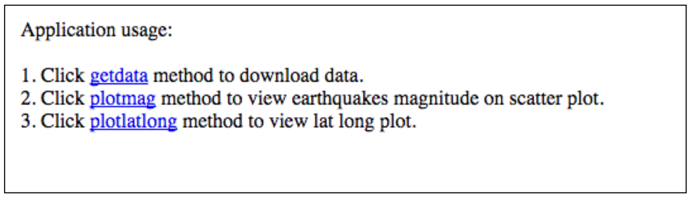
3.3. Deployment

Project is deployed using ansible. Ansible jobs are collected in a playbook and run on virtual clusters provided by Chameleon cloud. The tasks include cloning the git repository, installing software stack, installing dependencies and installing MongoDB.

3.4. Application

Application has a Cherrypy web server and python scripts that use the web server to send the output in the form of response so that users can see the output of the application in web browser.

Figure 1 shows default page showing application usage.



Application usage:

1. Click getdata method to download data.
2. Click plotmag method to view earthquakes magnitude on scatter plot.
3. Click plotlatlong method to view lat long plot.

Fig. 1. Application usage

3.5. Steps to Execute

Steps to execute the project are explained in detail in README.md file [3].

4. CLUSTERING ALGORITHMS

4.1. K-Means Clustering

Given a target number, k, of clusters to find, K-means algorithm locates the centers of each of those k clusters and the boundaries between them. It does this using the following algorithm [4].

- Step 1: Start with a randomly selected set of k centroids
- Step 2: Determine which observation is in which cluster, based on which centroid it is closest to (using the squared Euclidean distance.

$$\sum_{j=1}^{p}(x_{ij} - X_{i^1 j})^2$$

  where p is the number of dimensions)
- Re-calculate the centroids of each cluster by minimizing the squared Euclidean distance to each observation in the cluster
- Repeat 2. and 3. until the members of the clusters (and hence the positions of the centroids) no longer change.

4.2. DBSCAN

DBSCAN (Density-Based Spatial Clustering of Application with Noise) is a clustering algorithm. Given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), and marks points as outliers if they lie alone in low-density regions. Unlike K-Means Clustering, DBSCAN doesn't need to specify the number of clusters as it finds all the clusters that satisfy the requirement. Following points summarize the algorithm [5].

- Step 1: For each point in the data set, an n-dimensional sphere of radius $e$psilon is drawn around the point (if you have n-dimensional data).
- Step 2: If the number of points inside the sphere is larger than $m$in-samples, the center of the sphere is set as a cluster, and all the points within the sphere belong to this cluster.
- Step 3: Loop through all the points within the sphere with the above 2 steps, and expand the cluster whenever it satisfy the 2 rules.
- Step 4: For the points, which do not belong to any cluster, are treated as outliers.

5. EXECUTION PLAN

This is how the project is executed on week-by-week basis.

1. **13 Mar 2017 - 19 Mar 2017** Research on USGS data and fetching it. Deploy Mongo DB and Mongo Express tool to fetch data from Mongo DB. Data analysis and finalizing the analysis for project.
2. **20 Mar 2017 - 26 Mar 2017** Write script in Python for downloading USGS data at run-time and storing in local MongoDB. Research on clustering algorithms.
3. **27 Mar 2017 - 02 Apr 2017** Implement K-Means clustering and DBSCAN clustering in sample project. Understand the result. Create a VM on Chameleon.
4. **03 Apr 2017 - 09 Apr 2017** Write Python script to read data from MongoDB, implement K-Means clustering algorithm. Research on Plotly. Plot clustering result using Plotly.
5. **10 Apr 2017 - 16 Apr 2017** Implement DBSCAN clustering and plot clustering result using Plotly. Write ansible scripts for deployment.
6. **17 Apr 2017 - 23 Apr 2017** Deploy using ansible and run the project on various clouds. Observe and document benchmarks. Update and finalize the Project Report

6. BENCHMARKING

The performance of application was tested on the speed and latency while data input/output and data processing. Two different sized datasets were used for testing. They are given in Table 'Datasets'. The Table 'VM Configuration' shows details of VMs used for performance testing. Results are shown in Table 'Benchmark Results'.

Figure 2 shows time taken for data read and data processing with small dataset. Figure 3 shows the result with large dataset.

Performance tests are included in the python scripts themselves. As the scripts are executed, results are written in text files in 'benchmark' folder at project directory.

Table 1. VM Configuration

Cloud	Chameleon	Jetstream
Image	Ubuntu-Server-14.04-LTS	ubuntu-14.04-trusty-server-cloudimg
Group	pearth	pearth
Flavour	m1.small	m1.small
Assign Floating IP	True	True

Table 2. Datasets

Parameter	Dataset1	Dataset2
Region	North, South America	North, South America
Duration	2 years	2 years
Min Magnitude	4	2
Data Size	Small	Large

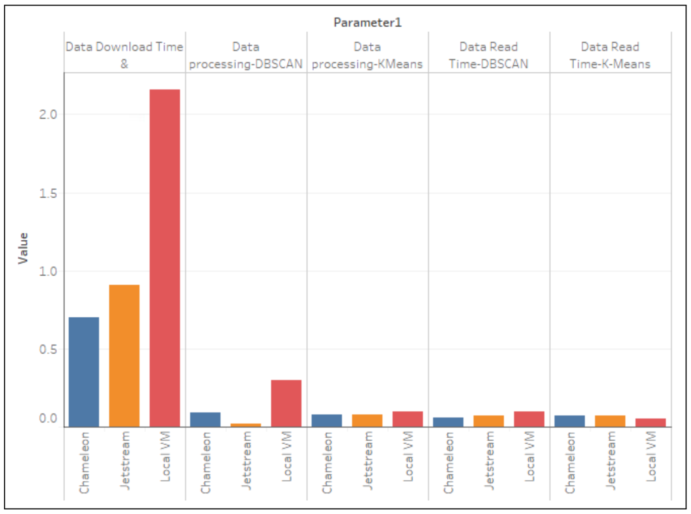


Fig. 2. Time taken for small dataset

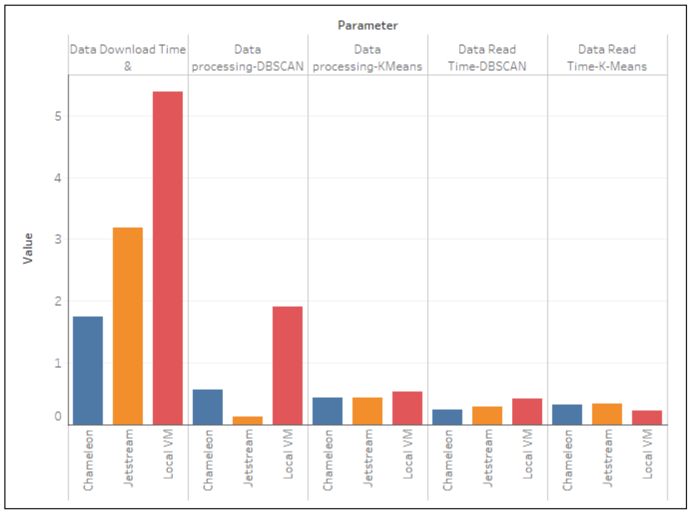


Fig. 3. Time taken for large dataset

7. RESULT AND ANALYSIS

TODO

8. CONCLUSION

This project was an opportunity to use Big Data open source software and projects. We can conclude that Ansible is a powerful tool for one click deployment and continuous integration. Cloudmesh client is a convenient tool to work with multiple cloud environments at the same time. Plotly allows creating interactive visualization.

While researching for the algorithms we came to know that DBSCAN is better choice for classification if one doesn't want to specify number of clusters beforehand.

Analysis of earthquakes data showed interesting facts. There were more than 7,000 earthquakes of magnitude more than 4 in 2 years span from yest 2015 till 2017. Frequent earthquakes occur in the west coast of North America.

9. ACKNOWLEDGEMENTS

This project is undertaken as part of the I524: Big Data and Open Source Software Projects course at Indiana University. The author would like to thank Prof. Gregor von Laszewski and his associates from the School of Informatics and Computing for providing all the technical support and assistance.

10. LICENSING

Project uses Apache license ver 2.0.

REFERENCES

[1] USGS, "Earthquake hazards program," Web Page. [Online]. Available: https://earthquake.usgs.gov/

[2] ——, "Search earthquake catalog," Web Page. [Online]. Available: https://earthquake.usgs.gov/earthquakes/search/

[3] N. Sathe, "Readme," Code Repository, April 2017, accessed: 2017-4-25. [Online]. Available: https://github.com/cloudmesh/earth/blob/master/README.md

[4] B. Govan, "Clustering using scikit-learn," Web Page, May 2013. [Online]. Available: http://fromdatawithlove.thegovans.us/2013/05/clustering-using-scikit-learn.html

[5] Q. Kong, "Clustering with dbscan," Web Page, August 2016. [Online]. Available: http://qingkaikong.blogspot.in/2016/08/clustering-with-dbscan.html

11. WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Nandita Sathe. She completed all the work related to development of this application including research, testing and writing the project report.