IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 9, No. 2, June 2020, pp. 269~275
ISSN: 2252-8938, DOI: 10.11591/ijai.v9.i2.pp269-275

STA/LTA trigger algorithm implementation on a seismological dataset using hadoop mapreduce

Youness Choubik[1], Abdelhak Mahmoudi[2], Mohammed Majid Himmi[3],Lahcen El Moudnib[4]
[1,3]LIMIARF Laboratory, Faculty of Sciences, Mohammed V University in Rabat, Morocco
[2]Ecole Normale Supérieure, Mohammed V University in Rabat, Morocco
[4]LGRN Laboratory, Scientific Institute, GEOPAC Research Center, Mohammed V University in Rabat, Morocco

Article Info

Article history:

Received Nov 24, 2019
Revised Apr 6, 2020
Accepted Apr 20, 2020

Keywords:

Big data
Hadoop
MapReduce
Seismic detection
STA/LTA

ABSTRACT

In this work we implemented STA/LTA trigger algorithm, which is widely used in seismic detection, using Hadoop MapReduce. This implementation allows to find out how effective it is in this type of tasks as well as to accelerate the detection process by reducing the processing time. We tested our implementation on a seismological dataset of 14 broadband seismic stations and compare it with the traditional one. The results show that MapReduce decreased the processing time by 34% compared to the traditional implementation.

Corresponding Author:

Youness Choubik,
LIMIARF Faculty of Sciences,
Mohammed V University in Rabat, Morocco.
Email: youness.choubik@gmail.com

1. INTRODUCTION

In recent years, processing data using traditional tools has become a difficult task due to the huge amount of available data. Hence the need of new tools and frameworks that facilitate and accelerate data processing. Big Data tools have become widely used in many fields, including Seismology. Where Given the Flexibility, Scalability and the ability of the Big Data tools to accelerate parallel processing of massive amounts of data, geophysicists became more and more dependent on these tools. For example, Addair et al. [1] cross correlated a global dataset consisting of over 300 million seismograms. This required 42 days using a conventional distributed cluster. By re-architecting the system to run as a series of MapReduce jobs on a Hadoop cluster they achieved a factor of 19 performance increase on a test dataset. Magana-zook et al. [2] used Hadoop and Spark to perform a large-scale calculation of seismic waveform quality metrics, and compared their performance with that of a traditional distributed implementation. They found that both Spark and MapReduce were about 15 times faster than the traditional distributed implementation. On the other hand, based on processing 43 TB using MapReduce and Spark, they predicted that for a dataset of 350 TB, Spark running on a 100 node cluster would be about 265 times faster than traditional implementation.

Mohammadpoor et al. [3] Conducted a comprehensive review on the application of Big Data analytics in oil and gas industry. Many research adopted hadoop to manage, store, and to analyze seismic data quickly [4-12]. Apache Spark is also used in many recent studies to analyze large volumes of seismic data [13-15]. Other researchers worked on Parallel algorithm to speed up their processing time [16-19].

Seismic detection is an important task in seismology, it allows for identifying seismic events, which permits deductions about the interior of the Earth, identify areas with high seismic activity and adding more seismic stations to better understand the seismicity in these areas. There are many seismic event detection algorithms [20], in this paper we are interested in applying the short term to long term trigger algorithm (STA/LTA) [21], which is widely used in seismic detection. We applied STA/LTA to the three-component records of the XB seismic network installed in Morocco between 2009 and 2013. Due to the large amounts of seismic data that we will be working on, we used Hadoop MapReduce to benefit from its power of processing data in a distributed fashion.

2. HADOOP FRAMEWORK

Apache Hadoop is an open-source framework that allows for distributed processing of large datasets across clusters of commodity hardware. Hadoop includes four principle components, Hadoop MapReduce for parallel processing of large data sets, Hadoop YARN for cluster resource management and job scheduling, Hadoop Distributed File System (HDFS) which is the data storage module and Hadoop Common that support the above Hadoop modules.

The Hadoop cluster is composed of two types of nodes, masters and slaves. The master nodes run the NameNode (NN), which manages the Hadoop Distributed File System (HDFS) by storing meta-data of files, managing the file system namespace, executing operations on files and directories such as accessing, closing and opening. On the other hand, the slave nodes run the DataNodes (DN), which performs read-write operations on the file systems according to the instructions of the NameNode as shown in Figure 1.

Hadoop processes data using MapReduce which is the programming paradigm that was first developed by Google [22]. MapReduce allows the processing of large amounts of data in a distributed and fault-tolerant manner. The MapReduce function operates on a set of <key, value> pairs and is composed of a map and a reduce part. In the map part a <key, value> pairs are passed to the function, and after doing the desired processing on those pairs, the function returns a new intermediate pairs. In the reduce part, the function takes a list of values per key as argument and returns another pairs after a summary operation. Between the map and the reduce functions another function is executed by Hadoop, which is the shuffle and sort function. In this function the values returned by the Map function which belong to the same key are assembled in one list and passed to the reduce function to be processed as shown in Figure 2. Hadoop MapReduce allows the processing of large volumes of data in a cost effective manner. Using Hadoop, even limited structures can easily access intensive computing capabilities.

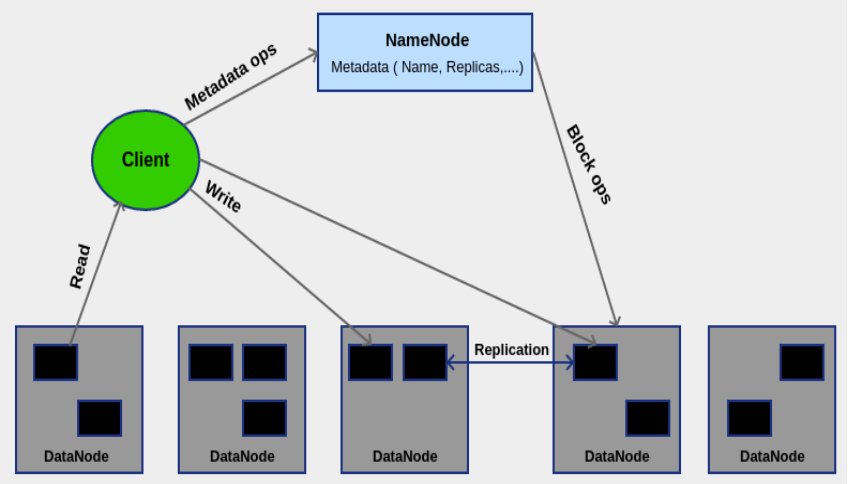

Figure 1. MapReduce architecture

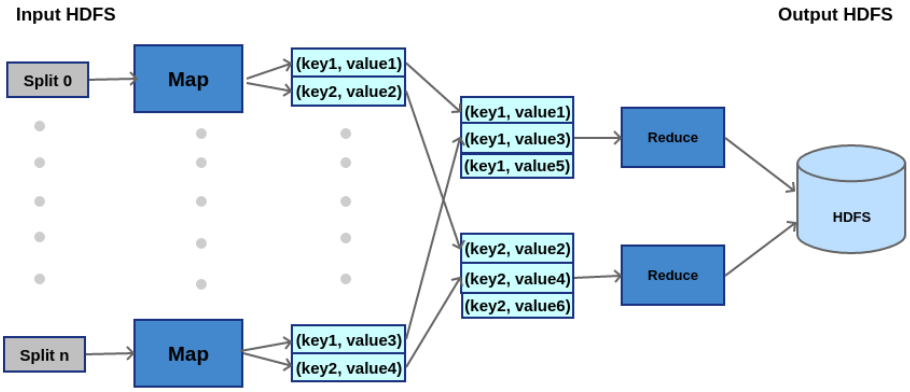

Figure 2. MapReduce architecture

3. SHORT TERM AVERAGE TO LONG TERM AVERAGE

The Short Term Average to Long Term Average (STA/LTA) is a trigger algorithm used for seismic detection. It is widely used in many processing software of the weak-motion seismic networks, as well as in portable seismic recorders, it may also be useful in many strong motion applications. It improves the detection of weak earthquakes and decreases the number of false detections triggered by seismic noise. A decreased number of false triggers and trigger's selectivity minimize the work of analysts [23].

The STA/LTA algorithm calculates the average of the absolute amplitude of a seismic signal in two consecutive moving-time windows as shown in Figure 3. The short time average (STA) represents the current average of a short duration over which an event could occur, while the long time average (LTA) represents the previous average of a longest duration to assess the seismic noise.

$$ST\ A/LT\ A > \alpha \quad (1)$$

$$ST\ A/LT\ A < \beta \quad (2)$$

The STA/LTA trigger algorithm requires three main parameters, the short term duration, the long term duration and the trigger threshold value α. When the current average is greater than the previous average, the ratio STA/LTA exceeds a preset threshold value α (1), and an event is declared. An other parameter can be added, which is the detrigger threshold. When the ratio STA/LTA falls below the detrigger threshold β (2), the end of the seismic event is pointed out. The STA/LTA trigger algorithm misses some true seismic events and detects some false events. However, it is suitable for detecting local, micro and distant earthquakes.

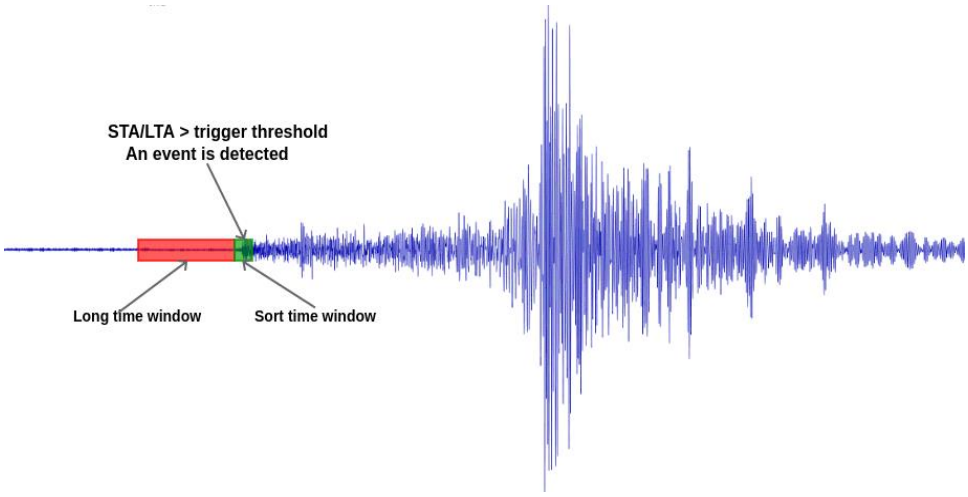

Figure 3. STA/LTA algorithm

4. METHOD

The dataset used in this work is from a seismological network called XB, a code assigned by the Federation of Digital Seismographic Networks (FDSN) archive to provide uniqueness to seismological data streams [24]. The XB seismological network was deployed in both Morocco and Spain, in the frame of the Project to Investigate Convective Alboran Sea System Overturn (PICASSO), between the time periods 2009 to 2013. It contained 93 seismic stations (labeled as PICASSO Spain (PS) and PICASSO Morocco (PM), of which 44 seismic stations were installed in Morocco. Figure 4 shows the positions of the XB network.

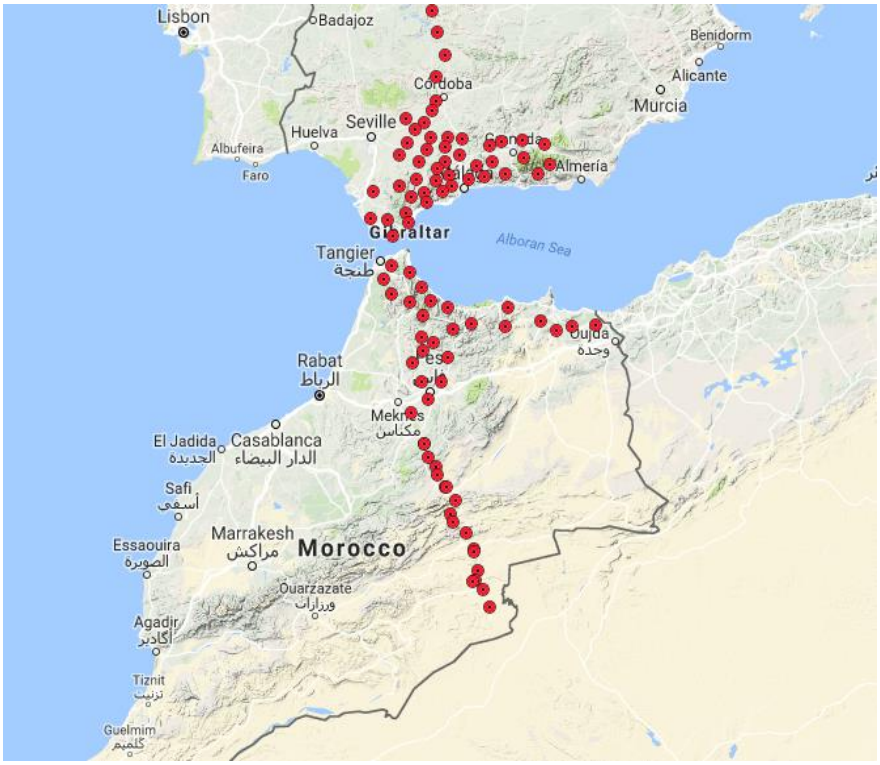

Figure 4. XB seismic network

The acquired time series data are generally of good quality due to the excellent seismological instrumentation deployed in a careful site location. The seismic data are SEED files (The Standard for the Exchange of Earthquake Data), obtained from the Incorporated Research Institutions for Seismology Data Management Center (IRIS DMC) and offering a large amounts of a continuous seismic data corresponding to many seismic stations around the world. SEED format is measured at one point in space and at equal intervals of time [25].

In order to use the downloaded SEED files in the Hadoop platform, we chose to use Apache Avro data serialization system which provides a compact, fast and binary data format. According to [2], AVRO provided the best API and portability across Big Data tools, as well as other features such as compact serialized size.

To test our implementation, we used a dataset of 14 stations from PM01 to PM16. The stations PM09 and PM10 were excluded from our study because of downloading problems. Figure 5 shows the size of the Avro files corresponding to every seismic station. We used Hadoop framework to process our dataset across a cluster of commodity hardware. The configurations of the cluster used in this work is given in Table 1.

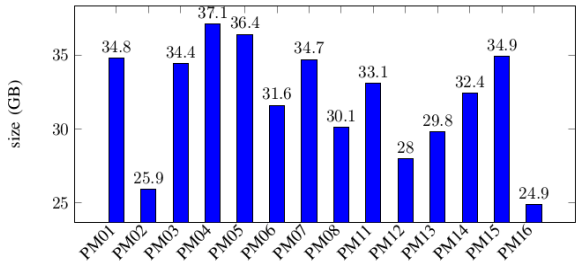

Figure 5. Size of avro files of the 14 PM stations

Table 1. Hadoop cluster configuration

	master nodes	worker nodes
Count	1	5
Storage	100 GB	900 GB
Memory	5 GB	25 GB
Cores	2	5

After configuring our cluster, the Avro files were sent to it using the Command Line Interface (CLI) and Hadoop takes care of storing and duplicating data into the different nodes. In order to find seismic events and the stations that simultaneously trigger them, we used two MapReduce functions. The first one for detecting seismic events at a given station by implementing the STA/LTA algorithm, while the second function will search for the events detected at the same time in more than one station.

In the first map function, we defined two windows, one for calculating the Short Time Average (STA window) and the other for the Long Time Average (LTA window).The two windows are slided by one sample and the average of each window is calculated, then the above averages are used to calculate the STA/LTA ratio. We consider an assumed seismic event in a single component when the ratio exceeds a defined trigger threshold. We save the time at which the event occurred and we continue sliding the windows and calculating the ratio until it falls below the detrigger threshold. At this moment we declare the end of the assumed seismic event as shown in Figure 6. Thus, the first map function returns the station name as a key and the value is a concatenation of the start time, end time and the channel name.

In the first Reduce function the list of value for each key (station) is sorted by the start time, then the whole list is treated so that if the event is detected in only one channel it is considered as a noise. Otherwise, if it is detected in more than one channel in the same station this event is considered as an assumed earthquake and the reduce function returns one as a key and a concatenation of the start time, end time, the channel and the station name as a value, as shown in Figure 7. Those <key, value> pairs will be treated by the second MapReduce function.

The second Map function just read then returns the data returned by first reduce function. The second Reduce function work on the output of the first reduce function, it returns the events detected in more than one station. The values returned by the first MapReduce function are sorted by detection time and each event is compared with the events that follow. When the interval of time between two events is lower than a defined value, the function check if the stations where the events occurred are neighbors. For that, a table of nearest neighbors is defined. That table contains the nearest neighbors for each station as shown in Figure 8.

```
Function map1(key, value):
    data = getDataFromBlock();
    initializ(Trigger_threshold,Detrigger_threshold,
    STA_window, LTA_window, STA_average,
    LTA_average, ratio=STA_average/LTA_average,
    channel, station)
    for x in data do
        update( STA_window, LTA_window,
        STA_average, LTA_average, ratio)
        if ratio ≥ Trigger_threshold then
            set(eventStart)
        end
        if ratio ≤ Detrigger_threshold then
            set(eventEnd)
            context.write(key=station,
            value=eventStart+eventEnd+channel);
        end
    end
end
```

Figure 6. The first map algorithm

```
Function map1(key, value):
    data = getDataFromBlock();
    initializ(Trigger_threshold,Detrigger_threshold,
    STA_window, LTA_window, STA_average,
    LTA_average, ratio=STA_average/LTA_average,
    channel, station)
    for x in data do
        update( STA_window, LTA_window,
        STA_average, LTA_average, ratio)
        if ratio ≥ Trigger_threshold then
            set(eventStart)
        end
        if ratio ≤ Detrigger_threshold then
            set(eventEnd)
            context.write(key=station,
            value=eventStart+eventEnd+channel);
        end
    end
end
```

Figure 7. The first reduce algorithm

```
Function reduce2(key, valueList):
    sort(valueList)
    x=0
    while x < valueList.size()-1 do
        t1 = valueList(x).geteventStart()
        value = valueList(x)
        count = 1
        y = x+1
        while y < valueList.size() do
            t2 = valueList(x).geteventStart()
            if t2-t1<delay then
                station1 = valueList(x).getstation()
                station2 = valueList(x).getstation()
                if areNeighbours(station1, station2) then
                    value += valueList(y)
                    count++
                end
            else
                x=y-1
                break
            end
        end
        if count > 1 then
            context.write(key=1, value=value+key)
        end
    end
end
```

Figure 8. The second reduce algorithm

5. RESULTS AND DISCUSSION

To test our implementation we used a dataset consisting of data from 14 stations. To find out the improvement realized by MapReduce we considered a traditional implementation as a reference and compared its results with that of MapReduce.

The time needed for processing data by the reference implementation was almost 13 hours and half, while MapReduce needed nearly 9 hours to accomplish the tests. The MapReduce implementation decreased the processing time by 34%. As the dataset become larger the factor will increase too. Parallel processing allows multiple processors to work on these divided tasks, so that they run entire programs in less time.

By applying the STA/LTA trigger algorithm to the seismic data corresponding to the XB seismological network in morocco, we were able to detect 199177 events in the first MapReduce function. By applying the second MapReduce, the number of events detected in more than one station decreased to 11513 events. Figure 9 shows the number of events in each seismic station.

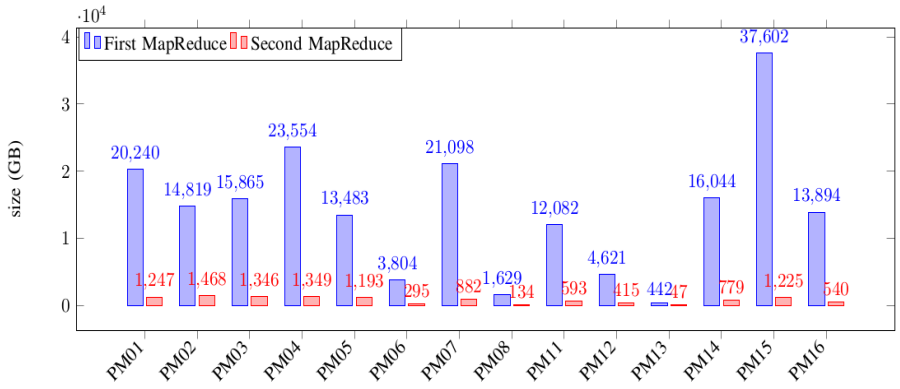

Figure 9. Number of seismic events detected per Station

6. CONCLUSIONS AND FUTURE WORK

We have shown in this paper the usability of Hadoop MapReduce for seismic detection, particularly using Short Term Average to Long Term Average (STA/LTA) algorithm. We compared MapReduce implementation performance with that of a traditional implementation. The results show that time needed for processing goes from almost 13 hours and half using the traditional implementation to nearly 9 hours by using MapReduce. So MapReduce decreases the processing time needed for processing large amount of seismic data.

Looking forward, our goal is to apply seismic detection on the entire XB network. This requires further optimizing since Short Term Average to Long Term Average lacks accuracy, and we should combine other techniques to improve its detection accuracy.

ACKNOWLEDGEMENTS

We acknowledge the Incorporated Research Institutions for Seismology (IRIS) (iris.edu) for providing the data used in this work. Our seismic data are obtained from the IRIS Data Management Center (DMC) (https://ds.iris.edu/ds/nodes/dmc)

REFERENCES

[1] Addair, Travis & Dodge, Douglas & Walter, William & Ruppert, Stanley. Large-Scale Seismic Signal Analysis with Hadoop. *Computers & Geosciences*. 66, 2014. 10.1016/j.cageo.2014.01.014.

[2] Magana-zook, S. & Gaylord, J.M. & Knapp, D.R. & Dodge, Douglas & Ruppert, Stanley. Large-scale seismic waveform quality metric calculation using Hadoop. *Computers & Geosciences*. 94, 2016. 10.1016/j.cageo.2016.05.012.

[3] Mohammadpoor, Mehdi & Torabi, Farshid. Big Data analytics in oil and gas industry: An emerging trend. Petroleum. 10.1016/j.petlm.2018.11.001, 2018.

[4] Rahim, Lukman & Kudiri, Krishna & Bahattacharjee, Shiladitya. Framework for parallelisation on big data. PLOS ONE. 14. e0214044. 10.1371/journal.pone.0214044, 2019.

[5] Javed, Shahrukh. Processing and Analysis of Large-Scale Seismic Signal in Hadoop Platform. Advances in Robotics & Mechanical Engineering. 1, 2018. 10.32474/ARME.2018.01.000103.
[6] Li, Zhenlong & Yang, Chaowei & Liu, Kai & Hu, Fei & Jin, Baoxuan. Automatic Scaling Hadoop in the Cloud for Efficient Process of Big Geospatial Data. *International Journal of Geo-Information*. 5, 2016. 10.3390/ijgi5100173.
[7] Zhonghua, Ma. Seismic data attribute extraction based on Hadoop platform. 180-184, 2017. 10.1109/ICCCBDA.2017.7951907.
[8] Mohammadzaheri, Afsaneh & Sadeghi, Hossein & Hosseini, Keivan & Navazandeh, Mahdi. DISRAY: A distributed ray tracing by map-reduce. Computers & Geosciences. 52. 453-458, 2012. 10.1016/j.cageo.2012.10.009.
[9] Yan, Xuesong & Zhu, Zhixin & Wu, Qinghua. Intelligent inversion method for pre-stack seismic big data based on MapReduce. Computers & Geosciences. 110, 2017. 10.1016/j.cageo.2017.10.002.
[10] Joshi, P. & Thapliyal, R. & Chittambakkam, A. & Ghosh, R. & Bhowmick, S. & Khan, S. Big Data Analytics for Micro-Seismic Monitoring. 10.4043/28381-MS, 2018.
[11] Chen, Zhuang & Zhang, Ti. Processing and analysis of seismic data in Hadoop platform. 1-5, 2017. 10.1109/CIACT.2017.7977288.
[12] Krauzowicz, Lukasz & Szostek, Kamil & Dwornik, Maciej & Oleksik, Pawel & Piórkowski, Adam. Numerical Calculations for Geophysics Inversion Problem Using Apache Hadoop Technology. 291. 440-447, 2012. 10.1007/978-3-642-31217-5_46.
[13] Huang, Lei & Dong, Xishuang & Clee, T. A scalable deep learning platform for identifying geologic features from seismic attributes. The Leading Edge. 36. 249-256, 2017. 10.1190/tle36030249.1.
[14] Yan, Yuzhong & Huang, Lei & Yi, Liqi. Is Apache Spark scalable to seismic data analytics and computations? 2036-2045, 2015. 10.1109/BigData.2015.7363985.
[15] Cortés, Gualberto & Morales-Esteban, Antonio & Shang, Xueyi & Martínez-Álvarez, Francisco. Earthquake Prediction in California Using Regression Algorithms and Cloud-based Big Data Infrastructure. *Computers & Geosciences. In press*. 10.1016/j.cageo.2017.10.011, 2017.
[16] Cattania, Camilla & Khalid, F. A parallel code to calculate rate-state seismicity evolution induced by time dependent, heterogeneous Coulomb stress changes. *Computers & Geosciences*. 94, 2016. 10.1016/j.cageo.2016.06.007.
[17] Chen, Po & Taylor, Nicholas & Dueker, Ken & Keifer, Ian & Wilson, Andra & McGuffy, Casey & Novitsky, Christopher & Spears, Alec & Holbrook, W. PSIN: A scalable, parallel algorithm for Seismic INterferometry of large-N ambient-noise data. *Computers & Geosciences*. 93, 2016. 10.1016/j.cageo.2016.05.003.
[18] Danek, Tomasz. Parallel and distributed seismic wave field modeling with combined Linux clusters and graphics processing units. 4. IV-208, 2009. 10.1109/IGARSS.2009.5417335.
[19] Kowal, A. & Piórkowski, Adam & Danek, Tomasz & Franczyk, Anna. Analysis of selected component technologies efficiency for parallel and distributed seismic wave field modeling. 10.1007/978-90-481-3658-2_62, 2010.
[20] Sharma, B. & Kumar, Amod & Murthy, V. Evaluation of seismic events detection algorithms. *Journal of the Geological Society of India*. 75. 533-538, 2010. 10.1007/s12594-010-0042-8.
[21] Houliston, D.J. & Waugh, G. & Laughlin, J. Automatic real-time event detection for seismic networks. *Computers & Geosciences*. 10. 431-436, 1984. 10.1016/0098-3004(84)90043-8.
[22] Dean, Jeffrey & Ghemawat, Sanjay. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*. 51. 137-150, 2004. 10.1145/1327452.1327492.
[23] Trnkoczy, A. Understanding and parameter setting of STA/LTA trigger algorithm. IASPEI New Manual of Seismological Observatory Practice. 2. 1-19, 2002.
[24] Alan Levander, Gene Humphreys, Pat Ryan. Program to Investigate Convective Alboran Sea System Overturn. International Federation of Digital Seismograph Networks. Dataset/Seismic Network. doi: https://doi.org/10.7914/SN/XB 2009, 2009.
[25] Ahern, T. K., R. Buland, and S. Halbert. SEED Format Version 2.4 Reference Manual, IRIS, 2009.