Big Data Analytics Project Proposal

A Fog Computing Prototype

Ali Alizadeh Mansouri
40102969

Marco Sassano
26658245

Abstract

Internet of Things (IoT) aims to bring every object (e.g. smart cameras, wearables, environmental sensors, home appliances, and vehicles) online, hence generating massive volumes of data that can overwhelm storage systems and data analytics applications. Cloud computing offers services at the infrastructure level that can scale to IoT storage and processing requirements. However, there are two major downsides to this two-layer infrastructure: The bottleneck is the network bandwidth, meaning that transmission of large volumes of raw data will oversaturate the network bandwidth on the way to the cloud, and there will be a high latency and response time for the results to come back from the cloud to the IoT devices. To overcome this limitation, Fog computing paradigm has been proposed, where cloud services are extended to the edge of the network to decrease the latency and network congestion. In our project, we implemented a Fog Computing infrastructure for early detection of epilepsy seizures using EEG timeseries data consisting of three layers: IoT devices and sensors, a Fog layer, and the Cloud. We performed the analytics on the 2nd and the 3rd layers. We report the methods, the results, and discuss the possible challenges, limitations, and future work.

1 Introduction

The number of Internet of Things (IoT) devices has increased to a great extent in recent years. It is estimated that 50 billion devices will be connected to the internet by 2020 [5]. On the other end side of the infrastructure, Cloud computing as a paradigm delivers computing services over the internet — the Cloud — to offer flexible resources to deal with a wide range of scalable computational demands. This includes analysis, aggregation, and storage of large volumes of data (Big Data) from the IoT devices. The total internet bandwidth crossing international borders in 2013 was 100 Tbps. Furthermore, while application demands are growing from 100s of terabytes towards petabytes per day, network capacity growth has been decelerating [12]. In other words, the bottleneck of such infrastructure lies on the network bandwidth between the IoT devices and the Cloud. This issue arises from the fact that most Cloud computing datacenters are geographically centralized, and situated far from the proximity of the end devices.

Fog Computing (FC) was proposed by Cisco in 2012 to address the needs of the applications which demand low latency and high response times [2]. As a distributed computing paradigm, FC acts as an intermediate layer in between Cloud services and IoT devices (or end users/devices in general) [9]. In this manner, the concept of FC is analogous to *data locality*, in which computational tasks are moved towards the data, instead of the other way.

Table 1: The breakdown of the EEG timeseries dataset used. Each record represents 178 tuples in a window of one second.

	Total	Positive	Negative
Training	7 000	1 392	5 608
Test	4 500	908	3 592
Total	11 500	2 300	9 200

1.1 Problem Specification

We consider the healthcare application of early detection of epilepsy seizures using EEG timeseries data, because such healthcare applications require close to real-time response times.

Our goal was to achieve less network congestion between the Fog and Cloud layers as well as higher response times for the EEG sensors. More specifically, we aim for a balanced tradeoff between fast and light-weight — even though less accurate — computations on the IoT side, and more accurate classifications but with higher latency on the cloud side.[1]

1.2 Related Work

FC in general as well as different applications requiring real-time response times have been considered vastly in the literature. For example, Tang *et al.* [11] implemented a hierarchical FC architecture for anomaly detection of pipe leakage in smart cities. We were inspired by the work of Diab Abdulgalil *et al.* [4], where they propose a FC architecture with SVM classification at the edge, and deep convolutional networks in the cloud. To the best of our knowledge, and to the date of this writing, this has been the only work which considered FC for efficient epilepsy seizures detection.

2 Materials and Methods

2.1 Dataset

Andrzejak *et al.* [1][2] provide a dataset of 500 individuals, each with 4097 data points for 23.5 seconds. This dataset is further reshaped and shuffled to 23 × 500 = 11500 records, where each record contains 178 data points (features) for 1 second by Qiuyi and Fokoue on UCI Machine Learning Repository [13]. Each record of this dataset is labeled with one of 5 classes, with one class being the seizure state. Since our goal was predicting the seizures, we further restructured this dataset into `(time,value)` tuples, and considered the seizure class of consecutive 178 tuples as positive, and all the groups of tuples belonging to other classes as negative. This resulted in 11500 × 178 = 2 047 000 tuples, which would be simulated as a stream of data generated by an EEG sensor. The final breakdown of the dataset is shown in table 1.

[1] *A note on the revision of the first two sections*: Compared to the project proposal, these sections have been mildly revised to 1) more accurately reflect the actual implementation of the project, and the materials — including the dataset used — and, 2) incorporate the professor's helpful comments from the project proposal.

[2] http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3&changelang=3

2.2 Project Structure

We will consider a three-tier architecture for our FC implementation, as described below:

1. **The sensor**: This layer consists of the EEG sensor, which generates streams of tuples in the form of `(time,value)`.
2. **The edge layer**: The edge layer[3] is the intermediate layer between the sensor and the cloud. Its job is to map and analyze the stream of tuples from the sensor.
3. **The cloud**: The cloud in a conventional FC architecture receives the preliminary results of the edge layer for further analysis and aggregation. However, we perform the analytics on the same dataset in the cloud in order to compare the performance of the two layers more accurately in terms of accuracy and resource consumption.

2.2.1 The Edge

At the time of this writing, there are only two frameworks dedicated to Fog/Edge Computing: Apache Edgent[4], and iFogSim [7][5] both of which are currently in a beta state. After an initial inspection, we chose Apache Edgent because it is in active development and more mature.

We implemented the sensor and the edge layers using Apache Edgent, which is a program model and runtime environment in Java suitable for stream processing. The sensor is implemented using a **Supply** class, whose job is to simulate generation of infinite streams of data from a variety of sources, including files, databases, network/internet resources, and physical sensors. In our case, we read the dataset, and output the tuples as a stream.

The edge layer consists of two parts:

- **Map**: As described previously, our goal was to come up with a fast and light-weight detection algorithm. Carney *et al.* [3] compared several epilepsy detection algorithms, from simple aggregation-style statistical measures, to more complicated machine learning algorithms such as deep convolutional networks. We experimented and compared the accuracy of the more simpler methods — to be more consistent with our goal — such as standardized moments (variance, skewness, kurtosis). Based on analysis of the dataset, we decided that the root of the sum of squared distances between every consecutive pair of tuples best describes the seizure classes of records. We denote this measure using $\mathscr{D}$, shown in equation 1. The $\mathscr{D}$ measure is implemented in the *map* step, which receives streams of tuples from the sensor, and maps them to $(\mathscr{D})$ tuples as another stream.

$$\mathscr{D} = \sqrt{\sum_{i=2}^{n} (X_i - X_{i-1})^2} \tag{1}$$

[3] It is worth mentioning that, Fog Computing is sometimes referred to as "Edge Computing" in the literature (e.g. [10, 6]). Certain surveys, on the other hand, treat the two as separate paradigms by considering Fog Computing as a generalization of Edge Computing, with Edge Computing representing a set of network devices (including the end devices) not necessarily interacting with the cloud [9]. For the sake of clarity, we will use "Fog Computing" to refer to the paradigm in the focus of this project, as suggested by [2] and [9].

[4] http://edgent.apache.org/

[5] https://github.com/Cloudslab/iFogSim

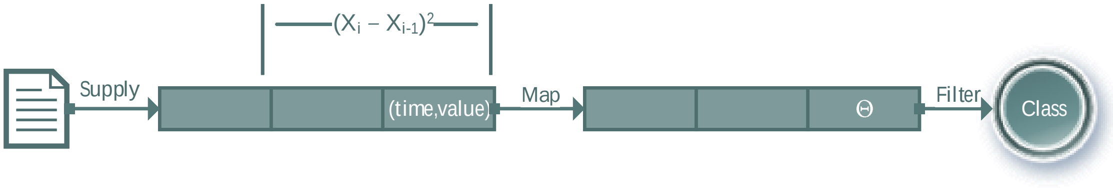


Figure 1: The implemented edge stream processing pipeline.

- **Analyze**: This step will classify every window of 178 tuples (equivalent to one second of data points) according to a threshold Θ. The threshold Θ is the determined optimal value among different $\mathscr{D}$ candidates obtained from the previous step on the training phase. Any values of $\mathscr{D} > \Theta$ will be classified and detected as an indication of seizure on the testing phase.

Figure 1 shows the described edge pipeline.

2.2.2 The Cloud

The *k-means* algorithm is mentioned as one of the epilepsy seizure detection methods by Carney *et al.* [3]. It is also the major method used for anomaly detection in the EGADS library by Yahoo [8][6]. Therefore, we implemented the cloud analytics step using the *k-means* algorithm, with the number of classes/clusters predetermined in our dataset. We implemented our algorithm using Python on Apache Spark (PySpark) as an efficient parallelized computing framework.

As the measure of distance, we experimented with the Euclidean distance. We also tried transforming data into 178 dimensions. However, we did not achieve a good accuracy due to, what we believe to be, different time intervals present in the dataset. We ultimately selected a distance measure similar to the edge layer, which is the sum of absolute consecutive pairwise differences for each window of 178 tuples, here denoted by $\mathscr{S}$ and shown in equation 2.

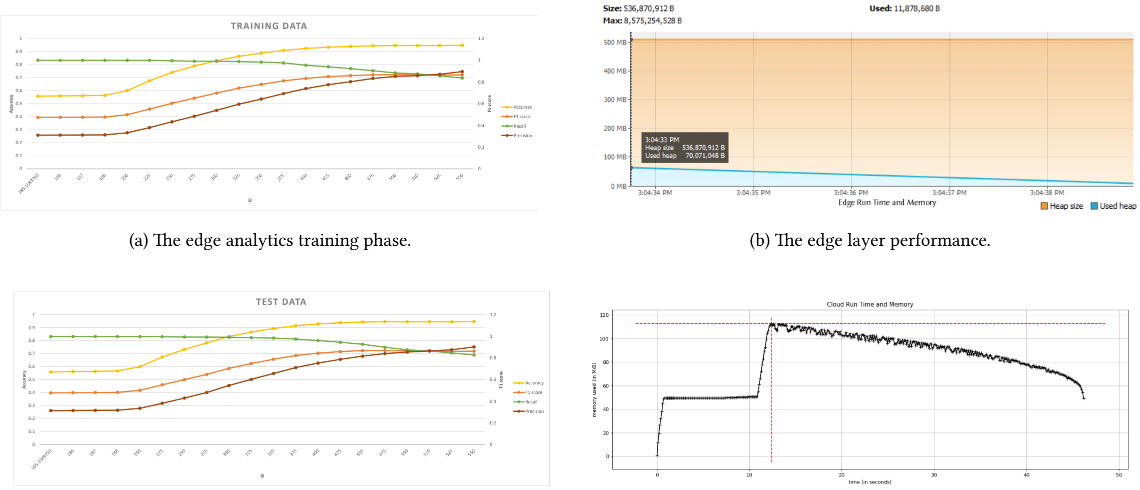
$$\mathscr{S} = \sum_{i=2}^{n} |X_i - X_{i-1}| \tag{2}$$

3 Results

In this section, we demonstrate the results from the analytics of the two layers presented in the previous section, and then compare the performance of the two. The experiments were performed on a PC with a Intel Xeon CPU E3-1271 v3 @ 3.6GHz, and 32GB of memory.

The accuracy, recall, precision, and F1 score for training and test sets for the edge analytics are shown in figures 2a and 2c, respectively. Based on the training results, we selected a value of Θ = 300 as an optimal balance between accuracy and recall, because in such an application we would rather have more false positives (false seizure alarms) than false negatives (missed seizure episodes).

[6] https://github.com/yahoo/egads



(a) The edge analytics training phase.

(b) The edge layer performance.

(c) The edge analytics training phase.

(d) The cloud layer performance.

Figure 2

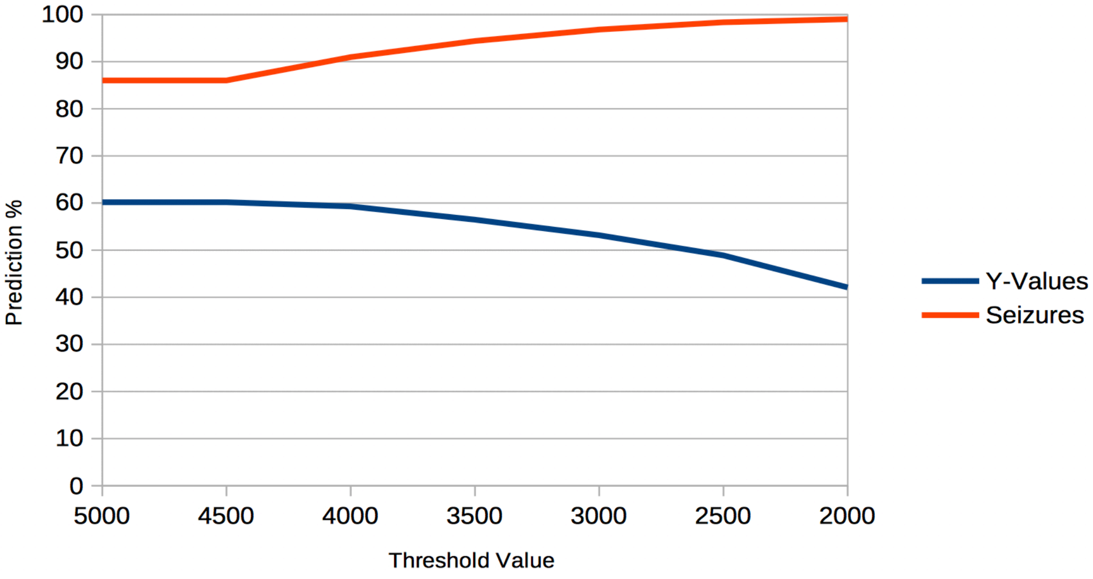


Figure 3: The threshold value of the distance measure versus accuracy for the k-means algorithm of the cloud layer.

The threshold value ($\Theta = 300$) of the distance measure ($\mathcal{S}$) in the k-means algorithm of the cloud layer versus accuracy for test data is displayed in figure 3. The results showed that we could achieve higher levels of accuracy as we decreased the threshold/cut-off value for $\mathcal{S}$ which determines assigning the records to clusters in the k-means algorithm. We selected 3500 as an optimal value based on the results of the training phase.

The runtime and memory consumption of the edge and cloud layers for a sample run are shown on figures 2b and 2d, respectively. The results show that the edge analytics consumes less memory (almost as half), and performs much faster (about $2s$ against about $45s$).

4 Discussion

The results from the Results section demonstrate the advantages of FC for applications which demand near real-time response times, such as those in healthcare.

Instead of having to deal with high accuracy but low latency, as is common on a conventional two-tier IoT/Cloud architecture, an FC architecture — such as one implemented in this project — allows us to come up with a balanced tradeoff between performance, latency, and accuracy. Our edge layer analytics solution has lower accuracy in the experiments ($\approx$ 80%), but it consumes much less resources than the cloud analytics solution, and achieves negligible latency. The cloud solution, on the other hand, achieves higher accuracy ($\approx$ 95%), but consumes more resources, and incurs a higher response time, considering it will be run in the cloud, farther from the sensors. That means, for example, an application implemented using this architecture can use the edge layer to perform fast though inaccurate analytics for early notification of an onset of epileptic seizures, and at the same time send the data (or an aggregated collection of data) to the cloud for more accurate, but slower, results. Therefore, we believe that we have achieved the desired goal for the project, which is a simple demonstration of obtaining the ability to select from two levels of tradeoffs, as mentioned above, via an architecture of FC.

During the course of the project, we faced certain challenges, some of which led us to make choices or forgo others stated in our proposal. For example, we planned to implement a light version of k-means or DBSCAN to analyze the streams; however, as the Apache Edgent is still incubating and lacks a complete documentation, setting it up consumed more time than expected. Moreover, we planned to set up the cloud instance on an IBM Watson IoT hub, and connect the Edgent to its API to send the streams directly to the cloud. This also became unrealistically difficult, because we learned that we needed to write an extra application besides the one in Apache Spark so that it would run in the IBM Streaming Analytics engine.

We believe that our work can take advantage of many improvements in the future. The first step is to take advantage of a publish-subscribe message broker, such as Apache Kafka or IBM Watson IoT Hub, to process the incoming streams of multiple sensors (or those processed by an edge node), and interact with the cloud. We would also like to try more advanced analytics solutions on the edge; however, the performance requirements of such algorithms and models must be studied and experimented to make them feasible to perform on the edge. Another step would be to receive the result of the analysis from the cloud, and actuate or act based on it on the edge or even the sensors. In the case of our healthcare application, for example, this could be a notification of the final verdict of the cloud on the early detection of a seizure. The cloud layer can also benefit from improvements. Other variants of k-means or other clustering methods could be compared based on accuracy and performance. While the lower response time of the cloud analytics is acceptable, faster responses are still more favorable in order to achieve higher user satisfaction.

References

[1] Ralph G. Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev. E*, 64:061907, Nov 2001.

[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[3] Paul R Carney, Stephen Myers, and James D Geyer. Seizure prediction: methods. *Epilepsy & behavior : E&B*, 22 Suppl 1(Suppl 1):S94–S101, dec 2011.

[4] Huda Diab Abdulgalil. *A Multi-Tier Distributed fog-based Architecture for Early Prediction of Epileptic Seizures*. PhD thesis, University of Waterloo, 2018.

[5] D Evans. The internet of things: How the next evolution of the internet is changing everything. Technical report, 2011.

[6] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev.*, 45(5):37–42, sep 2015.

[7] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[8] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1939–1947. ACM, 2015.

[9] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog Computing: A Taxonomy, Survey and Future Directions. In Beniamino Di Martino, Kuan-Ching Li, Laurence T Yang, and Antonio Esposito, editors, *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, pages 103–130. Springer Singapore, Singapore, 2018.

[10] W Shi, J Cao, Q Zhang, Y Li, and L Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[11] Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, and Qing Yang. A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart Cities. In *Proceedings of the ASE BigData &#38; SocialInformatics 2015*, ASE BD&#38;SI '15, pages 28:1—-28:6, New York, NY, USA, 2015. ACM.

[12] Ashish Vulimiri, Carlo Curino, P Brighten Godfrey, Thomas Jungblut, Jitu Padhye, and George Varghese. Global Analytics in the Face of Bandwidth and Regulatory Constraints. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 323–336, Oakland, CA, 2015. {USENIX} Association.

[13] Qiuyi Wu and Ernest Fokoue. Epileptic seizure recognition data set. https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition. Accessed: 2019-04-15.