An End to End Real Time Architecture for Analyzing and Clustering Time Series Data: Case of an Energy Management System

Hanaa Talei*
School of Science and Engineering
AlAkhawayn University in Ifrane
Ifrane, Morocco
H.Talei@aui.ma

Mohamed Essaaidi
ENSIAS School
Rabat, Morocco
essaaidi@gmail.com

Driss Benhaddou
Engineering Technology Department
University of Houston
Houston, TX, USA
dbenhaddou@central.uh.edu

Abstract— Big data is a field that fascinated many researchers from different areas to study intelligent and robust techniques to analyze extremely large data sets, reveal patterns about human behaviors and then make important decisions such as predicting the next human activity. Smart grid is a cyber physical system that aims at updating the current old-fashion electrical grid by incorporating the latest ICTs to improve the generation, the distribution and the consumption of electricity. The use of sensors in smart grids becomes crucial as it allows an energy management system to analyze massive generated sensory data and use machine learning algorithms to take advantage of the customer's participation to reduce the cost of power. However, big data field revealed a very long list of tools to analyze data either in real time or batch modes so the decision of what tools to use for a particular case becomes a challenging one. The purpose of this paper is to present an end- to – end architecture for a real time test bed implemented at Al Akhawayn University in Ifrane Morocco to analyze and cluster time series sensor data using an IoT architecture composed of Kaa (a middleware), Kafka (a real-time data messaging queue), Spark (an in-memory data analytics platform) and k-means (a clustering algorithm).

Keywords— Smart Grids, IoT, Big Data, time series, Spark, k-means

I. INTRODUCTION

New ICTs have been incorporated and revolutionized many old systems such as Internet, the latter is shifting toward a new internet model called internet of things where computing devices are attached to daily objects allowing them to send and receive data and be monitored and controlled remotely[1]. Likewise, researchers are working toward modernizing the electrical grid by integrating new technologies guarantying a steady power supply, a low CO_2 emission system, a non-failure grid (no hazard blackouts) and most importantly involving the customer in reducing the cost of power. The need of an intelligent energy management system becomes a must as the latter will analyze a vast amount to data generated from different sources (sensors, smart meters...) and take important decisions to optimize the grid operations.

The data generated by sensors networks will reveals important information such as consumption and behavior patterns and by applying machine learning algorithms we can achieve significant energy and budgetary savings. However one challenge about the use of sensor network is the vast generated data that requires robust computing platform for analysis: big data analytics. The purpose of this paper is to present a test bed architecture implemented at Al Akhawayn University in Ifrane (AUI) to cluster energy consumption data extracted from times series generated by wireless sensors; the clustered data will be used by an energy management system to control the energy consumption in the university campus which fits in the strategic plan of the institution to reduce energy to 50% by 2020.

978-1-7281-1182-7/18/$31.00 ©2018 IEEE

The paper is structured as follows: Section II gives a general overview about the test bed architecture. Section III discusses the choice of the middleware used. A description of the real time pipeline used is presented in Section IV. Section V covers the choice of Spark as an analytical platform while section VI discusses the choice of k-means as a clustering algorithm and finally conclusions and future work are presented in Section VII.

II. AUI Test bed: a General Architecture

With a world wild increase in electricity demand, smart grids emerged as a solution for various problems within the current electrical grid. Some novel aspects about smart grids are the use of distributed energy resources such as renewable energy sources, energy storage, and electrical Vehicle as well as involving the user in the decision and the control of energy usage through demand load management. Microgrids are building blocks of smart grids and the future power grid will be composed of interconnected Microgrids similar to how the Internet is a mesh of networks[2]. A typical example of a Microgrid is a university campus where consumption can be efficiently controlled by using an energy management system presented (EMS) in figure1. Clearly an EMS will have to process vast amount of data generated from different sources to monitor and control a Microgrid. From a customer side, an EMS will use the data generated by sensors and after being processed, it will provide useful information such as energy consumption, market prices allowing the user to control appliances remotely and hence save energy and budget[3].

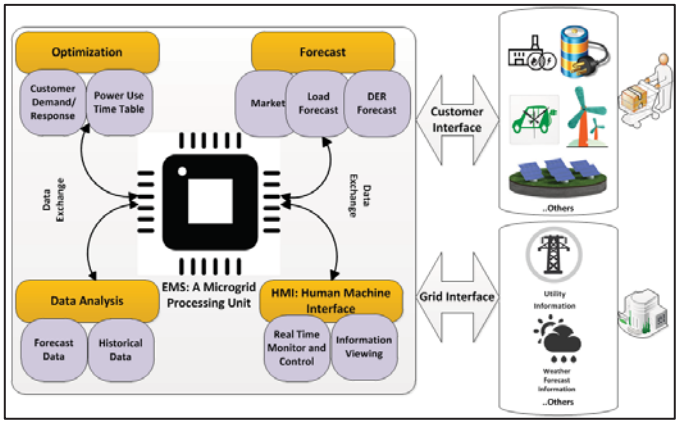

Figure 1: Energy Management System - A General Architecture

To analyze time series generated by multiple sensors, we designed a 4 layers architecture depicted in figure 2. The first layer consists of a middleware used to hide the heterogeneity of the data received from multiple sources (structured, semi structured and unstructured data) and inject it into the data pipeline (the next layer) in a "standard" format as to be processed by the next layer. Given that we are implementing a real time system, the generated data is not saved into a database (such as Cassandra) but rather placed in a low latency messaging system as to be analyzed by a big data platform. The most important layer in our architecture is the data analytics one as data will be consumed from the messaging queue, transformed into RDDs and then clustered to unveil the hidden consumption pattern. The latter is achieved using k-means, a clustering algorithm incorporated in the machine library of Apache Spark. A detailed description of the tools chosen in our experiments is presented in the next sections.

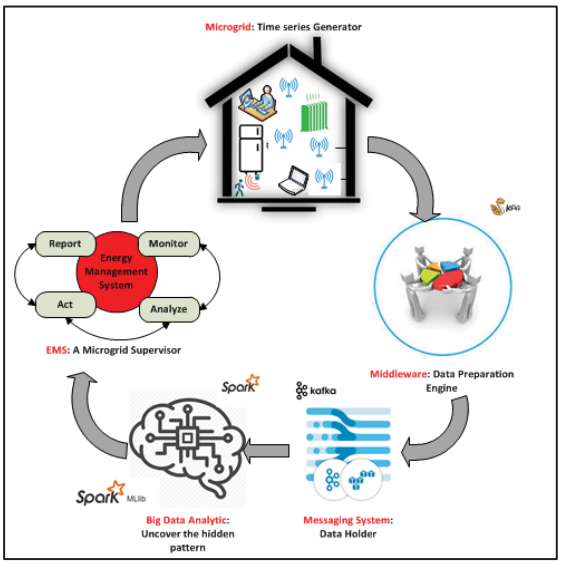

Figure 2: An End to End 4 Layers Architecture to Analyze Time Series in Real Time

III. The choice of an IoT middleware: Kaa

Internet of things (IoT) is a network of devices (cars, washing machines, heaters, computers...) equipped with electronic devices cable of sending data periodically and communicating with other devices with a minimum human interaction. Like an operating system, the needs of an interface between the hardware and applications becomes a prerequisite: a middleware. In [4] and [5], the authors described a middleware as a system that embraces the heterogeneity of IoT devices and also supports the essential ingredients of composition, adaptability, and security aspects of an IoT system. The choice of a Kaa as a cloud based IoT middleware was based on a literature review carried in [6].

Kaa is a free, highly flexible, multi-purpose, 100% open-source middleware platform for implementing

complete end-to-end IoT solutions. Kaa middleware consists of:

- Kaa Server: used for administrative capabilities such as managing tenants, applications, devices and users
- Kaa Extensions: used to improve the platform functionalities
- Endpoints SDKs: To facilitate the creation of clients' applications to run on different connected devices, Kaa provides a library that provides client APIs (in different programming languages such as C, Java and C++) for various Kaa platform features and handle data communication

It worth noting that Kaa can be used as a Sandbox (used for proof of concept), as a single node (a preconfigured package to start the application), as a cluster nodes (a minimum of three Linux nodes for a distributed system) as it can also be deployed in AWS[7]. For scalability purpose, we used Kaa 10.0 as a cluster of 3 Linux nodes using the architecture shown in figure 3.

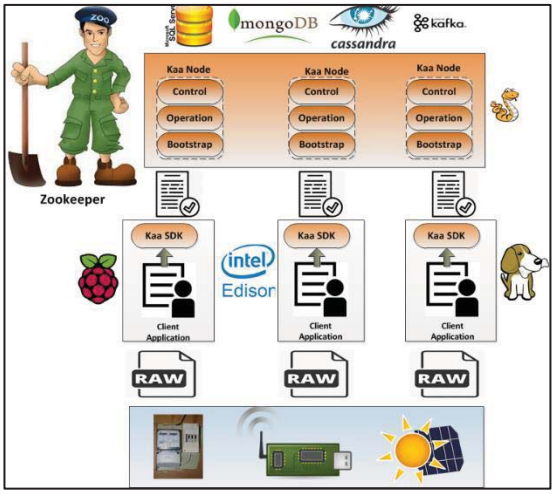

Figure 3: Kaa Middleware as a Cluster of 3 Nodes

Each node consists of a Kaa server called Kaa node which offers three services: bootstrap service (sends the information to the endpoints about operations services connection parameters such as IP, TCP port…), operation service (communicates with multiple endpoints concurrently), and control service (manages overall system data such as the available operations by receiving the information from Zookeeper, provide web based interface for the user to manage user accounts, application data…..). One important component in the architecture is Apache Zookeeper which is primary used to reliably coordinate Kaa cluster by actively balancing the load between the three nodes.

Kaa middleware also offer the possibility of transferring the generated logs by the operation service into a database or a messaging queue such as flume or Kafka. The choice of the log appender used in our experiments in explained in the next section.

IV. The choice of a distributed Messaging System: Kafka

Given the huge amount of data generated by different devices such as sensors, a real time data processing architecture is in need of a low latency queue where data will be hosted for a short period of time and then transferred to another application: a messaging system. There are two main types of messaging systems: point to point (where data in the queue can be fetched by one consumer only) and publish-subscribe (where messages are persisted in a topic and consumers subscribe to one or more topic then consume all the message in the topics); however, most of the messaging patterns will follow the second category.

Apache Kafka is a distributed publish-subscribe messaging system that was developed at LinkedIn and later on became a part of Apache project. Kafka can host a high volume of data as it performs 2 million writes/sec which fits perfectly with real time streaming data analytics applications. In [8] and [9] the authors compared Kafka to other messaging system and Kafka proved to have a better throughput, a built-in partitioning, a replication and inherent fault-tolerance, which makes it a good candidate for large-scale message processing applications.

Figure 4 presents Kafka architecture we used in the test-bed:

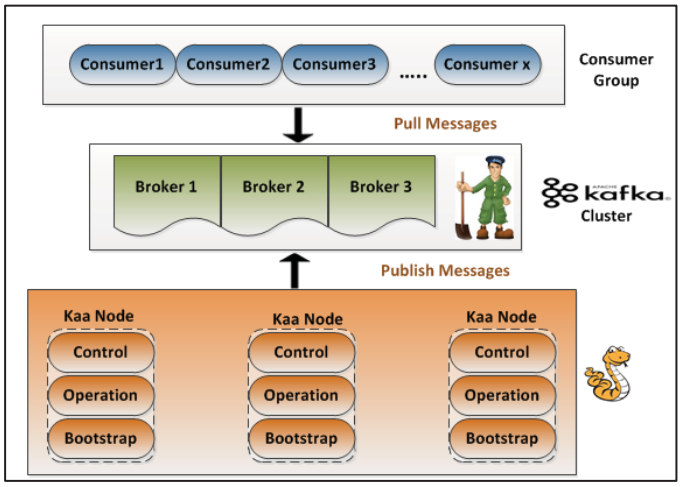

Figure 4: Kafka Cluster Architecture

To maintain a scalable, distributed system, ensuring load balancing we are using a Kafka cluster which consists of three brokers (Kafka Servers) that act as agents between the

producers (Kaa nodes) and the consumers (data analytics tools). To manage and coordinate the cluster, Apache Zookeeper is primary used to notify the producers/consumers about the presence or the failure of a broker as it also take care of choosing a broker leader by election. The next section will describe how messages generated (published) by Kaa nodes will be consumed by a real time data analytic tool called Spark.

V. The choice of an Analytic Engine: Spark

Lately, researchers become more motivated to work on analyzing big data and use machine learning algorithms to extract valuable information that can be used to build a model to prove or controvert some scientific hypothesis. As there are different types of problems so there are different analysis techniques categories; this includes:

- **Classification**: the goal is to predict the category of the input data. For instance predicting if the weather will be rainy, sunny...
- **Regression**: the model has to predict a numerical value rather than a category. For example predicting the electricity consumption per day.
- **Clustering**: is about organizing similar items into groups; this includes grouping university members into students, stuff, faculty…
- **Graph analytics**: when data can be transformed into graph with nodes and links then graph analytics come to place to find relationship between entities such in social networking.
- **Association Analysis**: the goal is to find some rules to capture associations between events. For instance during exam weeks if a customer will go to a supermarket to buy some school supplies then probably he/she will need to buy some vitamins so putting these two items together will increase the sales.

Given the key characteristics of big data that are volume, variety and veracity, analyzing this vast amount of data become a real challenge as we need powerful tools used in this context. Two main analytic tools are used in this field Hadoop and Spark; but one has to decide about which framework is better depending on the problem to solve. The work carried in [10] presents a performance comparison between Spark and Hadoop based on three criteria: execution time, throughout and speedup and results proved that Spark is better in dealing with huge amount of data but it requires higher memory allocation. As we are interested in building a real time architecture where data will be processed with a minimum delay we opted for Spark as a big data analytics ecosystem.

Spark was initiated at UC Berkeley in 2009 and was transferred to Apache foundation in 2013 and since then it fascinated many contributors worldwide. Spark provides a very expressive programming model with more than 20 highly fast/efficient in-memory distributed operations or transformations with very few lines of code which makes it a very good candidate to consider for iterative applications. Another Spark advantage is its ability to support both batch and streaming processing which perfectly fits the case of an Energy management system presented in figure 1; by looking at “data analysis” section, the latter will process forecast data (streaming) and historical data (batch). Spark also provides simple APIs for Scala, Java, Python and SQL programming using an interactive shell to accomplish analytical tasks using external and built in libraries[11].

The spark layer diagram or stack consists of components built on top of Spark computational engine presented in figure 5:

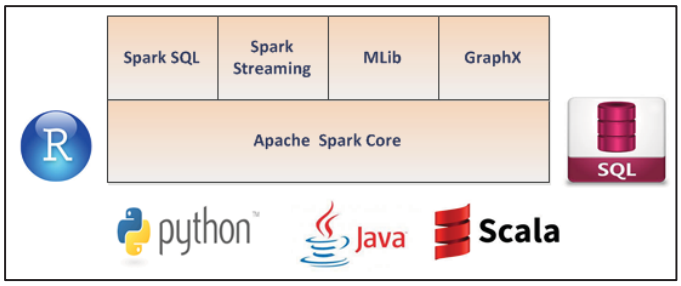

Figure 5: Apache Spark Components

The first layer, Apache Spark core, is where the core capabilities of Spark are implemented; this includes support for distributed scheduling, memory management and fault tolerance. The core is also responsible about the interaction with different schedulers such as YARN (a resource scheduler used in the Hadoop ecosystem), Mesos (the first open-source cluster manager that handles workloads in a distributed system) and various no SQL systems like Hbase. Spark Core also contains a set of APIs for defining RDDs (Resilient Data Sets) which are the main programming abstraction in Spark as it dispatch data across many computing nodes in parallel and transform it. SparkSQL allow querying structured and unstructured data through a common query language by connecting to many data sources. It contains APIs to convert query results into RDDs in Python, Scala and Java programming. Spark Streaming is responsible of aggregating data (micro batches) coming from streaming data ingestion systems and convert them to RDDs. MLib is Spark scalable library used to evaluate models using machine learning algorithms with APIs in Java, Scala, R and Python. For the last Spark component, GraphX, is a library for graph analytics that allow the conversion of vertex-edge model to RDDs and it contains a package of algorithms used for graph analytics[11].

Given the hardware/software requirements to process big data, we work with a commodity cluster made of 1 master (master) and 3 worker nodes to execute the jobs. To sum up, the reason why we used Spark as a data analytics tool is that it provides a diverse interactive, management and analysis of data in a low processing time compared to Hadoop. In the next section we discuss in details the use of Spark MLib component and the choice of using k-means as an algorithm to cluster time series data.

VI. The choice of a Clustering Algorithm: k-means

Clustering is one of the key unsupervised machine learning techniques used to group data into sets and then discover some high facts and knowledge using the discovered patterns about the data sets with the same behavior[12]. Figure 6 presents our testbed general architecture. Our main goal is to group the time series data generated by sensors periodically, which consists of electricity consumption in a particular building, during a particular day and in a specific time slot. Having grouped the data in a set of clusters, we can reveal important information such identifying the buildings that consume more electricity during a particular day/week/season, the electricity consumption distribution per week/month/year, predicting the buildings consumption based on their function (students dormitories building, computing center building, academic building….) and many other patterns after analyzing the generated clusters.

In [13], authors carried a survey of clustering algorithms for big data and categorized them into five classes:

1. **Partition based**: where data are divided into partitions where each partition represent a cluster (example k-Means discussed next).
2. **Hierarchical based:** where data are organized in a hierarchical tress depending on the proximity where a tree diagram represent a dataset and leaves represent data. Unlike portioning based clustering where cluster content can change iteratively, in a hierarchical clustering once a split or merge is executed, it cannot be undone (example BIRCH, balanced iterative reducing and clustering using hierarchies).
3. **Density based:** it groups together data that have many nearby neighbors, therefore outliers will reside on a low density region (example DBSCAN, Density-based spatial clustering of applications with noise).
4. **Grid based:** is a type of clustering algorithm that partition the dataset into a finite number of cells to form a grid structure and then find the dense regions (clusters) in the grid (example OptiGrid, optimal grid).
5. **Model based:** is a special data mining clustering algorithm that assumes that data is generated by a model (probability distribution) and tries to uncover the original model from the data (example COBWEB).

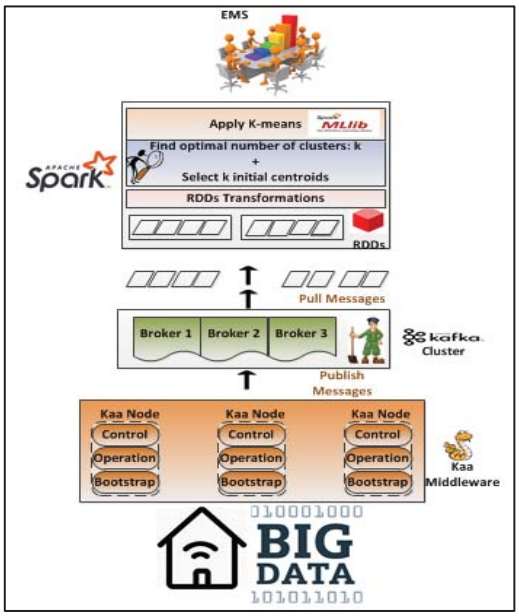

Figure 6: Al Akhawayn Detailed Architecture

It is important to mention that one key feature to consider in big data solutions is to choose simple algorithms (as complex algorithms will increase the computational time) and in [13]and [14], authors carried an empirical analysis of clustering algorithms for big data and experiments proved that for numerical data (such as power consumption), k-means, as a simple algorithm, showed excellent performance.

k-means is a simple yet an effective clustering algorithm that follows the following steps:

1. Select k initial centroids (cluster centers)
2. Repeat
   - Assign each data sample to the closest centroid
   - Calculate mean of cluster to determine new centroid
3. Until stopping criterion is reached
   - No changes to centroids
   - Number of sample changing clusters is below a certain threshold [15]

There are two main challenges about using k-means: the choice of initial centroids and the choice of the number of clusters. Many approaches have been used to select the initial

centroids in k-means, but the easiest way is to run k-means multiple times with different random centroids and choose the best results[16]. As for the number of clusters, in [17] and [18], analyzed some cluster validity indices (CVI) used to approximate the optimal number of clusters with an emphasis on two VCIs (Silhouette and Dunn) used to handle large amount of data in a low computational time and experiments showed excellent performance outputs.

k-means algorithm is one of the oldest but most commonly used algorithms given its simple implementation[19]. Figure 7 presents different terminologies related to k-means algorithm. The key feature of k-means is to group data into clusters (figure 7 shows 3 different ones) so that there is a high intra-cluster similarity (average distance between a point and its cluster centroid), low inter-cluster (average distance between every cluster centroid and the global one) similarity and unveil patterns from the grouped objects. In our experiments, we used spark MLib as a platform for big data analysis that comes with a package of machine learning tools for classification, regression, dimension reduction, clustering and rule extraction[20].

As K-mean's first steps is to choose a set of centers from the data set, this can lead sometimes to wrong clustering results as two random center initializations can get easily two centers into the same cluster. Spark Mlib offers the option of distributed processing using a parallelized version of the k-means++ algorithm called k-means|| particularly used for big data clustering[21]. K-means++ is considered as an initialization algorithm for K-means as it result in a set of initial centers which is close to the optimum solution[22].

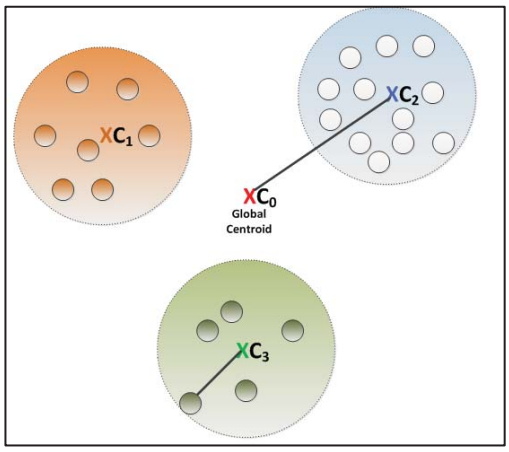

Figure 7: Data Clustering Using K-means

VII. Conclusion & Future Work

In the last few years, the amount of data generated in different fields (web, medicine, business…) increased considerably and researchers are eager to make use of the produced data to reveal patterns mainly for economical profits.

Traditional techniques are not efficient given the volume/velocity/variety of big data that's why the use of appropriate big data tools becomes a must. The list of big data analytics tools is a very long one and one gets really confused to choose the appropriate tools for a particular case. The purpose of this paper is to present a real time architecture to analyze sensory time series data about university campus electricity consumption; the architecture is a four layers one which consists of a:

- Kaa middleware: an open source and free application used to hide the heterogeneity of the underlying IoT system as well as an interface between hardware and the other applications.
- Kafka messaging system: a distributed/ low latency streaming platform where other applications such as Spark can fetch data with high throughput.
- Spark as processing engine: we eliminated Hadoop as Spark can be 100x faster for large scale datasets by using RDDs, in-memory computing, a package of transformations functions, and the use of parallel programming.
- k-means as a clustering algorithm: the most well know clustering algorithm given its simple implementation and low computation time. The grouped/clustered data will be used by an energy management system to reveal important electricity patterns and hence optimize the Microgrid electricity usage both economically and electrically.

The goal of this paper is to present the architecture used at al Akhawayn University smart grid project to cluster sensors data as a way to unveil some patterns that can be used by an energy management system to control the energy consumption within the university campus. To generate real "big data" sets from sensors in terms of volume, we need to run the experiments for years and then we can judge the effectiveness of the system. Currently we are populating Spark with a set of records collected from university electricity meters in the past year and we will reveal the obtained patterns and results in the next paper.

REFERENCES

[1] F. Silva and B. Herrera, "Big Data Analytics in IOT : Challenges , Open Research Issues and Tools," vol. 3, no. c, pp. 775–788.

[2] H. Talei, B. Zizi, M. R. Abid, D. Benhaddou, and N. Khalil, "Smart Campus Microgrid : Advantages and the Main Architectural Components," 2015.

[3] H. Talei, "Smart Campus Energy Management System : Advantages , Architectures , and the Impact of using Cloud Computing," in ICSDE '17, 2017.

[4] M. Elkhodr, S. Shahrestani, and H. Cheung, "A middleware of the Internet of Things," Int. J. Comput. Networks Commun., vol. 8, no. 2, pp. 159–178, 2016.

[5] E. I. Datateknik, "A comparative study of open- source IoT middleware platforms . En jämförande studie av open- source IoT middleware plattformar .," EXAMENSARBETE INOM DATATEKNIK, GRUNDNIVÅ, 2018.

[6] P. P. Ray, "A survey of IoT cloud platforms," Futur. Comput. Informatics J., vol. 1, no. 1–2, pp. 35–46, 2017.

[7] "Architecture overview · Kaa." [Online]. Available: https://kaaproject.github.io/kaa/docs/v0.10.0/Architecture-overview/. [Accessed: 31-Jul-2018].

[8] V. John and X. Liu, "A Survey of Distributed Message Broker Queues," arXiv Prepr. arXiv1704.00411.

[9] J. Kreps, L. Corp, and L. Corp, "Kafka : a Distributed Messaging System for Log Processing," NetDB'11,Athens, Greece.

[10] Y. Samadi, M. Zbakh, and T. Claude, "Performance comparison between Hadoop and Spark frameworks using HiBench benchmarks," Concurr. Comput. Pr. Exper. 2017, no. November 2016, pp. 1–13, 2017.

[11] "Overview - Spark 2.3.1 Documentation." [Online]. Available: https://spark.apache.org/docs/latest/. [Accessed: 31-Jul-2018].

[12] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," Ann. Data Sci., vol. 2, no. 2, pp. 165–193, 2015.

[13] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "IEEE TRANSACTIONS ON A Survey of Clustering Algorithms for Big Data : Taxonomy and Empirical Analysis," vol. 2, no. 3, 2014.

[14] R. Ding, Q. Wang, Y. Dang, Q. Fu, H. Zhang, and D. Zhang, "YADING : Fast Clustering of Large-Scale Time Series Data," in Proceedings of the VLDB Endowment, 2015, vol. 8, no. 5, pp. 473–484.

[15] "K-Means Algorithm - Unsupervised Learning | Coursera." [Online]. Available: https://www.coursera.org/lecture/machine-learning/k-means-algorithm-93VPG. [Accessed: 31-Jul-2018].

[16] T. M. P. A. Saradha, "An Optimized repartitioned K-means Cluster algorithm using MapReduce Techniques for Big Data analysis," in International Journal of Advance Engineering and Research Development, 2017, pp. 157–165.

[17] J. M. L. J. García-gutiérrez, M. M. José, and C. R. Santos, "An approach to validity indices for clustering techniques in Big Data," Prog. Artif. Intell., 2017.

[18] S. Singh, "Big Data Mining of Energy Time Series for Behavioral Analytics and Energy Consumption Forecasting," in Energies 2018, 2018.

[19] "Most Popular Clustering Algorithms Used In Machine Learning." [Online]. Available: https://www.analyticsindiamag.com/most-popular-clustering-algorithms-used-in-machine-learning/. [Accessed: 01-Nov-2018].

[20] M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, "Big Data Machine Learning using Apache Spark MLlib," pp. 1–7, 2017.

[21] B. Bahmani, R. Kumar, and S. Vassilvitskii, "Scalable K-Means ++," pp. 622–633.

[22] M. Capó, A. Pérez, and J. A. Lozano, "An efficient approximation to the K-means clustering for massive data," Knowledge-Based Syst., vol. 117, no. 8, pp. 56–69, 2017.