Improving the Efficiency of Implementing K-Means Algorithm on Different Big Data Platforms

Manal A. Abdel-Fattah
Information Systems Department
Faculty of Computers and Information, Helwan University
Cairo, Egypt
manal_8@hotmail.com

Yehia M. Helmy
Business Information Systems Department
Faculty of Commerce and Business Administration, Helwan University
Cairo, Egypt
ymhelmy@yahoo.com

Sara M. Mosaad
Business Information Systems Department
Faculty of Commerce and Business Administration, Helwan University
Cairo, Egypt
sara.mosaad87@gmail.com

Abstract— Due to the expansion of the data size and the limitation of a single machine, considering parallelism in a distributed computational environment was a natural solution to overcome this expansion. Spark and Flink are open source frameworks for processing data in both real-time mode and batch mode. It provides several benefits such as fault-tolerant and large-scale computation which is a general engine for large-scale data processing. In this paper, we major in the efficiency optimization of one of the implicit partitioning clustering algorithm, K-means, to get high clustering performance by handling the indication of the optimal number of clusters issue based on a joint combination of three different methods.

Index Terms— k-means algorithm; Big Data Platforms; Spark; Flink; Number of Clusters Determination.

1 Introduction

In this section the main description of the sciences included in the thesis will be introduced which is about the Big Data different definitions and it's Six V's, the K-means Algorithm and its points of strength and weakness, The Big Data platforms; Apache Spark and Apache Flink and their architecture and reasons of choosing them and the properties of clustering algorithm that should be considered for big data analysis.

Big Data

According to Gartner, Inc. big data is defined as; "Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making." Similarly, TechAmerica Foundation

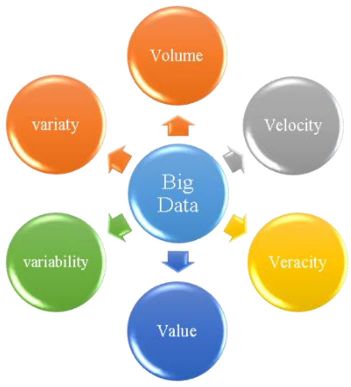

Fig.1: The Six V's of Big Data.

defines big data as follows; "Big data is a term that describes large volumes of high velocity, complex and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information."

The big data is not only characterized by the three Vs mentioned above but may also extend to six Vs, namely, volume, variety, velocity, variability, veracity and value . This 6V definition is

widely recognized because it highlights the meaning and necessity of big data (Gandomi and Haider 2015)(Hashem et al. 2015).
Volume. refers to the magnitude of data. Big data sizes are reported in multiple terabytes and petabytes. A survey conducted by IBM in mid-2012 revealed that just over half of the 1144 respondents considered datasets over one terabyte to be big data Definitions of big data volumes are relative and vary by factors, such as time and the type of data. What may be deemed big data today may not meet the threshold in the future because storage capacities will increase, allowing even bigger data sets to be captured.
Variety. refers to the structural heterogeneity in a dataset. Technological advances allow firms to use various types of structured, semi-structured, and unstructured data. Structured data, which constitutes only 5% of all existing data, refers to the tabular data found in spreadsheets or relational databases. Text, images, audio, and video are examples of unstructured data, which sometimes lack the structural organization required by machines for analysis. Spanning a continuum between fully structured and unstructured data, the format of semi-structured data does not conform to strict standards. Extensible Markup Language (XML), a textual language for exchanging data on the Web, is a typical example of semi-structured data.
Velocity. refers to the rate at which data are generated and the speed at which it should be analyzed and acted upon. The proliferation of digital devices such as smart phones and sensors has led to an unprecedented rate of data creation and is driving a growing need for real-time analytics and evidence-based planning. Even conventional retailers are generating high-frequency data. Wal-Mart, for instance, processes more than one million transactions per hour. The data emanating from mobile devices and flowing through mobile apps produces torrents of information that can be used to generate real-time, personalized offers for everyday customers. This data provides sound information about customers, such as geospatial location, demographics, and past buying patterns, which can be analyzed in real time to create real customer value.
Veracity. IBM coined Veracity as the fourth V, which represents the unreliability inherent in some sources of data. For example, customer sentiments in social media are uncertain in nature, since they entail human judgment. Yet they contain valuable information. Thus the need to deal with imprecise and uncertain data is another facet of big data, which is addressed using tools and analytics developed for management and mining of uncertain data.
Variability (and complexity). SAS introduced Variability and Complexity as two additional dimensions of big data. Variability refers to the variation in the data flow rates. Often, big data velocity is not consistent and has periodic peaks and troughs. Complexity refers to the fact that big data are generated through a myriad of sources. This imposes a critical challenge: the need to connect, match, cleanse and transform data received from different sources.
Value. Oracle introduced Value as a defining attribute of big data. Based on Oracle's definition, big data are often characterized by relatively "low value density". That is, the data received in the original form usually has a low value relative to its volume. However, a high value can be obtained by analyzing large volumes of such data.

Apache Spark

Spark is a next generation paradigm for big data processing developed by researchers at the University of California at Berkeley. It is an alternative to Hadoop which is designed to overcome the disk I/O limitations and improve the performance of earlier systems. The major feature of Spark that makes it unique is its ability to perform in-memory computations. It allows the data to be cached in memory, thus eliminating the Hadoop's disk overhead limitation for iterative tasks. Spark is a general engine for large-scale data processing that supports Java, Scala and Python and for certain tasks it is tested to be up to 100× faster than Hadoop MapReduce when the data can fit in the memory, and up to 10× faster when data resides on the disk. It can run on Hadoop Yarn manager and can read data from HDFS. This makes it extremely versatile to run on different systems (Maheshwar, Donavalli, and Bhovi 2017).
Apache Spark is user friendly programming interface used to decrease coding efforts and provide better performance in a majority of the cases with problems related to big data also it is a powerful processing framework that provides an ease of use tool for efficient analytics of heterogeneous data. Spark not just provides an alternative to Map Reduce, but also has options for SQL like querying with Shark and a machine learning library called MlLib. Apache Spark started as a research project at UC Berkeley in the AMPLab, was started with a goal to design a programming model that supports a much wider class of applications than MapReduce, while maintaining its automatic fault tolerance (Gopalani and Arora 2015).
A key concept of Spark is Resilient Distributed Datasets (RDDs). An RDD is basically an immutable collection of objects spread across a Spark cluster. In Spark, there are two types of operations on RDDs: (1) transformations and (2) actions. Transformations consist in the creation of new RDDs from existing ones using functions like map, filter, union and join. Actions consist of final result of RDD computations.
A Spark cluster is based on a master/slave architecture with three main components (Inoubli et al. 2018):
Driver Program: this component represents the slave node in a Spark cluster. It maintains an object called sparkContext that manages and supervises running applications.
Cluster Manager: this component is responsible for orchestrating the workflow of application assigned by Driver **Program to workers**. It also controls and supervises all resources in the cluster and returns their state to the Driver Program
Worker Nodes: each Worker Node represents a container of one operation during the execution of a Spark program.

Spark offers several Application Programming Interfaces (APIs) (Inoubli et al. 2018):
Spark Core: Spark Core is the underlying general execution engine for the Spark platform. All other functionalities and extensions are built on top of it. Spark Core provides in-memory computing capabilities and a generalized execution model to support a wide variety of applications, as well as Java, Scala, and Python APIs for ease of development.
Spark Streaming: Spark Streaming enables powerful interactive and analytical applications across both streaming and historical

data, while inheriting Spark's ease of use and fault tolerance characteristics. It can be used with a wide variety of popular data sources including HDFS, Flume, Kafka, and Twitter

Spark SQL: Spark offers a range of features to structure data retrieved from several sources. It allows subsequently to manipulate them using the SQL language.

Spark MLLib: Spark provides a scalable machine learning library that delivers both high-quality algorithms (e.g., multiple iterations to increase accuracy) and high speed (up to 100x faster than MapReduce).

GraphX: GraphX is a Spark API for graph parallel computation (e.g., PageRank algorithm and collaborative filtering).

Reasons to choose Apache Spark

There are many reasons for choosing Apache Spark to implement the enhanced k-means algorithm on it, these reasons are(Gopalani and Arora 2015):

1. Spark uses the concept of RDD which allows us to store data in memory and persist it as per the requirements. This allows a massive increase in batch processing job performance (up to ten to hundred times as much as that of conventional Map Reduce).
2. Spark also allows us to cache the data in memory, which is beneficial in case of iterative algorithms such as those used in machine learning.
3. Traditional MapReduce and DAG engines are suboptimal for these applications because they are based on acyclic data flow: an application has to run as a series of distinct jobs, each of which reads data from stable storage (e.g. a distributed file system) and writes it back to stable storage. They incur significant cost loading the data on each step and writing it back to replicated storage.
4. Spark allows us to perform stream processing with large input data and deal with only a chunk of data on the fly. This can also be used for online machine learning, and is highly appropriate for use cases with a requirement for real time analysis which happens to be an almost ubiquitous requirement in the industry.
5. In particular, MapReduce is inefficient for multi-pass applications that require low-latency data sharing across multiple parallel operations. These applications are quite common in analytics, and include:
   - Iterative algorithms, including many machine learning algorithms and graph algorithms like PageRank.
   - Interactive data mining, where a user would like to load data into RAM across a cluster and query it repeatedly.
   - Streaming applications that maintain aggregate state over time.

Apache Flink

Flink is an open source framework for processing data in both real-time mode and batch mode. It provides several benefits such as fault-tolerant and large-scale computation. The programming model of Flink is similar to that one of MapReduce. By contrast to MapReduce, Flink offers additional high-level functions such as join, filter and aggregation. Flink allows iterative processing and real time computation on stream data collected by different tools such as Flume and Kafka . It offers several APIs on a more abstract level allowing the user to launch distributed computation in a transparent and easy way. Flink ML is a machine learning library that provides a wide range of learning algorithms to create fast and scalable Big Data applications (Inoubli et al. 2018)

Flink system consists of several layers. In the highest layer, users can submit their programs written in Java or Scala. User programs are then converted by the Flink compiler to DAGs. A DAG produced by the Flink compiler is received by the Flink optimizer in order to improve performance by optimizing the DAG (e.g., re-ordering of the operations). The second layer of Flink is the cluster manager which is responsible for planning tasks, monitoring the status of jobs and resource management. The lowest layer is the storage layer that ensures storage of the data to multiple destinations such as HDFS and local files(Inoubli et al. 2018).

Reasons to choose Apache Flink

There are many reasons for choosing Apache Flink to implement the enhanced k-means algorithm on it, these reasons are [1][2]:

1. Apache Flink is an open source platform for distributed data processing which enables the user to write programs that can be distributed over number of worker nodes. This makes it possible to process large-scale datasets faster than a single computer could.
2. Flink’s API offers two dedicated iteration operators to specify iterations: 1) bulk iterations, which are conceptually similar to loop unrolling, and 2) delta iterations, a special case of incremental iterations in which the solution set is modified by the step function instead of a full recomputation. Delta iterations can significantly speed up certain algorithms because the work in each iteration decreases as the number of iterations goes on.
3. An important reason relates to iterations handling. Spark implements iterations as regular for-loops and executes them by loop unrolling. This means that for each iteration a new set of tasks/operators is scheduled and executed. Each iteration operates on the result of the previous iteration which is held in memory. Flink executes iterations as cyclic data flows. This means that a data flow program (and all its operators) is scheduled just once and the data is fed back from the tail of an iteration to its head. Basically, data is flowing in cycles around the operators within an iteration. Since operators are just scheduled once, they can maintain a state over all iterations.
4. Among the big data frameworks, we have mentioned earlier, Apache Flink and Spark are popular and efficient frameworks as these two have been used heavily in machine learning applications
5. owing to having personalize machine learning libraries called FlinkML and MLib, respectively. Because of the in-memory nature of the computations, we can argue that Flink and Spark provide a comparable feature set for machine learning applications. Even though these frameworks have been positioned as enablers of large-scale machine learning

.

2 Related Work

Cui et al. presents **an Optimized big data K-means clustering using MapReduce** which is an optimized K-Means for big data using MapReduce that eliminates the dependence of iteration and reduces the computation cost of algorithms and propose strategies for center merging, where a sequence of three MapReduce (MR) jobs are used for the purpose. However, sampling technique is used in the first MR job and in the final MR job then data set is mapped to centroids using the Voronoi diagram [3].

Weizhong Zhao et al. presents **Parallel k-means clustering based on MapReduce** but the initial seed selection is random. Moreover, the algorithm is iterative, and MapReduce jobs are influenced by multiple iterations [4].

Bahmani et al. presents a **Scalable K-Means++** which is an efficient parallel version k-means|| of the inherently sequential k-means++. The algorithm is simple and embarrassingly parallel and hence admits easy realization in any parallel computational model. The non-trivial analysis show that k-means|| achieves a constant factor approximation to the optimum and the experimental results on large real-world datasets demonstrate the scalability of k-means|| [5].

Xiao Cai et al. presents **Multi-View K-Means Clustering on Big Data** that is used to tackle the large-scale multi-view clustering problems with large scale heterogeneous data sets, all the methods are repeated 50 times using random initialization to report the average performance [6].

Sinha et al. presents **A Novel K-Means based Clustering Algorithm for Big Data** that is implemented on Spark and automate the input of number of clusters in advance, also it tackle the resolution problem and the experimental results prove that the algorithm outperforms the K-Means algorithm implemented in Spark Machine Learning Library and works efficiently on large scale data sets but data sets do not contain large scale real time streaming data and did not consider other important factors of big data such as velocity and veracity. Moreover, the algorithm scales gracefully on increasing the data size and adding more machines to cluster [7].

3 Proposed Work

3.1. An overview of K-means Algorithm

K-means clustering algorithm is a kind of partitioning clustering methods, a typical distance-based clustering algorithm, using distance as similarity evaluation. K-means algorithm is simple for large-scale data mining with high efficiency and scalability and fast with a more intuitive geometric meaning. It has been widely used in pattern recognition, image processing and computer vision. At the same time, the satisfactory results are obtained [8]. It performs the following steps to form the clusters [9]

K-means Algorithm

Input: k: the number of clusters or groups **D:**data set of 'n' objects
Output: Formed k clusters.
Algorithm:

```
1. Input k value and dataset.
2. If k == 1, then Exit.
3. Else
4. Select k objects from D to the closest centroid.
5. Assign each point $d_1$ in D to the closest centroid.
6. Calculate and update new cluster centroids.
7. Repeat from step 5 until centroids no longer move.
```

Here are the strength and weakness points of the k-means algorithm [10][11]:

Strength of k-Means algorithm:

i. Relatively efficient O(knt) where k is the number of clusters, n is the number of objects, t is the number of iteration.
ii. Very easy to implement and understand.
iii. Objects automatically assigns to its clusters.
iv. Often terminate at local optimum.

Weakness of k-Means algorithm:

i. The number of cluster, K, must be determined beforehand. So the user need to specify k (number of clusters)
ii. We never know the real cluster, using the same data, if it is inputted in a different way may produce different cluster if the number of data is a few. So different initial k objects lead to different clustering results.
iii. Unable to handle noisy data and outlier.
iv. Not suitable for non-convex shapes.
v. Cannot be applied directly to categorical data only numerical data.
vi. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
vii. We never know which attribute contributes more to the grouping process since we assume that each attribute has the same weight.

So in order to overcome those weaknesses is to use K-mean clustering if there are available many data and to reach a way to get the optimal number of clusters for a dataset and how to select the initial centroids in an optimal way.

3.2. Proposed Optimal Number of Clusters Indication Algorithm

The proposed algorithm is based on a joint combination of three different methods; Jump Statistics (JS), Gap Statistics (GS), A Fisher-wise criterion / Calinski and Harabasz (CH), which lies under variance Based Approach for indicating the optimal number of clusters.

The rationale beyond taking the results of those three methods

• *Manal A. Abdel-Fattah is currently working as an Associate Professor in Information Systems Department, Helwan University, Egypt, E-mail: manal_8@hotmail.com*
• *Yehia M. Helmy is currently working as a professor and head of business Information Systems Department, Helwan University, Egypt, E-mail: ymhelmy@yahoo.com*
• *Sara M. Mosaad is currently working as a Senior Teaching Assistant in business Information Systems Department, Helwan University, Egypt, E-mail: sara.mosaad87@gmail.com*

is that; One method is not sufficient enough to give the accurate results regarding the optimal number of clusters on a particular data set. So, here we have used these three important methods by taking majority must be granted logic to get the accurate results. If we include all the methods of determining the optimal number of clusters the complexity of the optimal number of cluster detection method will increase. These three methods are effective and qualitative to predict the number of clusters.

Optimal Number of Clusters Indication Algorithm

```
Input: Data Set
Output: Optimal Number of Clusters ($Opt_k$)
    1. Find the values of JS[i], GS[i] and CH[i], for i = 1,2,3,….k (number of cluster).
    2. Find $Z_{JS}$ = max [JS[i]] , find the value of i (number of clusters) = $N_{JS}$ for which JS [i] is maximum. Similarly find $Z_{GS}$ = min[GS[i]] , find the value of i = $N_{GS}$ for which GS [i] is minimum, find $Z_{CH}$ = max[CH[i]], find the value of i = $N_{CH}$ for which CH [i] is maximum.
    3. Apply majority must be granted logic (voting approach) among these three parameters denoted as $N_{JS}$, $N_{GS}$, $N_{CH}$, if at least 2 results in the same number of clusters then go to step 5, otherwise go to the next step.
    4. Let Calinski-Harabasz index $CH_i$ be the quality index for the value attained in step 2., then calculate the instantaneous rate of change between each two successive number of clusters r = $CH_{i+1} – CH_i$. find the maximum r, then the optimal number of cluster is i that maximize r.
    5. Stop.
```

3.3. Proposed K-means Algorithm

The proposed k-means is a hybrid algorithm that combines both; the proposed optimal number of clusters indication algorithm and the original k-means algorithm.

Proposed K-means Algorithm

```
Input: k: the number of clusters or groups D:data set of 'n' objects
Output: Formed k clusters.
Algorithm:
    1. Input k value and dataset.
        1.1. Find the values of JS[i], GS[i] and CH[i], for i = 1,2,3,….k (number of cluster).
        1.2. Find $Z_{JS}$ = max [JS[i]] , find the value of i (number of clusters) = $N_{JS}$ for which JS[i] is maximum. Similarly find $Z_{GS}$ = min[GS[i]] , find the value of i = $N_{GS}$ for which GS[i] is minimum, find $Z_{CH}$ = max[CH[i]], find the value of i = $N_{CH}$ for which CH[i] is maximum.
        1.3. Apply majority must be granted logic (voting approach) among these three parameters denoted as $N_{JS}$, $N_{GS}$, $N_{CH}$, if at least 2 results in the same number of clusters then go to step 5, otherwise go to the next step.
        1.4. Let Calinski-Harabasz index CHi be the quality index for the value attained in step 2., then calculate the instantaneous rate of change between each two successive number of clusters r = $CH_{i+1} – CH_i$. find the maximum r, then the optimal number of cluster is i that maximize r.
        1.5. Stop.
    2. If k == 1, then Exit.
    3. Else
    4. Select k objects from D to the closest centroid.
    5. Assign each point $d_1$ in D to the closest centroid.
    6. Calculate and update new cluster centroids.
    7. Repeat from step 5 until centroids no longer move.
```

4 Experimental Plan

The proposed algorithm will be simulated on a cluster of 4 machines with the configuration stated in Table 1. All the nodes were connected by a 100 Mbps Ethernet switch with UBUNTU 14.04 LTS operating system. The Hadoop version Hadoop 2.6.0 will be installed and on top of spark 1.6.0. Apache Hadoop, Spark and Flink are open source platforms that can be downloaded from [12], [13] and [14] respectively.

TABLE 1: Cluster configuration

	Master	**Node01**	**Node02**	**Node03**
CPU Configuration	Server machine, ML 350E, Gen 8, Intel Xeon CPU E5-24070 @2.20 GHz	Server machine, ML 350E, Gen 8. Intel Xeon CPU E5-24070 @2.20 GHz	Blade Server, HP BL 4603, Gen 9, Intel Xeon CPU E5-2630v3 @2.40 GHZ	PC, Intel, Core(TM), i7-4510U CPU GHz
CPU cores allocated to Spark	1	1	7	3
RAM allocated to Spark	10GB	10GB	10GB	6GB

4.1. Datasets

For checking the quality of our Spark/ Flink based clustering algorithm. We used synthetic large scale data sets to check the scalability. The data sets is considered as big data carrying all the big data characteristics to judge the quality of clusters. It is to be noted that the data source used is HDFS and the block size is 64MB. The data sets are thus stored across the various nodes in the cluster. To show the efficiency of Spark and Flink validity index, discussed later will be used.

4.2. Performance Evaluation

Our proposed algorithm has been compared with the clustering algorithm present in Spark MLLib library. Most of the distributed clustering algorithm present in literature is based on MapReduce, and as illustrated earlier performance of Spark is much better than MapReduce. Hence, experimental comparison with MapReduce based algorithm is not justified. First we will check our algorithm for the optimal number of clusters to be generated. For this we will use data set that cover all the big data characteristics. The data source used in both the cases is HDFS. the time taken to run our proposed algorithm and the K-Means given in MLLib will be measured, the execution time by varying the number of machines will be measured. However, for the basic K-Means the number of iteration required to get the optimal number of cluster will be measured, and whether the proposed algorithm can handle the over-resolution problem or not.

4.3. Cluster Validity

The performance of clustering algorithm will be measured with the validation indices. Calinski-Harabasz index is a widely used validation metric is used to assess the quality of the clusters because of its faster computation than other indices. The maximum value of the CH the better is the cluster quality, as it maximizes the dispersion between clusters and minimize dispersion within clusters. CH(C) is defined as follows:

$$CH(c)=\frac{trace(B_m)}{trace(W_m)}\times\frac{N-c}{c-1} \qquad (1)$$

Where Bm is the between-cluster scatter matrix, While Wm is the internal scatter matrix, Nis the total number of clustered samples and c is the number clusters.

$$W_m=\sum_{i=1}^{c} \sum_{x\in Ci} (x-ci)\,(x-ci)^T \qquad (2)$$

$$B_m=\sum_{i} ni\,(ci-k)(ci-k)^T \qquad (3)$$

Where,

- Ci are the set of points in the cluster.
- ci *is* the center of the cluster Ci,
- n_i is the number of points of Cluster Ci.
- k is the center of the input data set.
- k is the number of clusters
- c_m is the centroid of cluster m
- σ_m is the average distance of all points in the cluster m to centroid $c_{m.}$

5 Conclusion & Future Work

In this paper, we have presented an enhanced K-Means algorithm that will be implemented on both Apache Spark and Apache Flink big data platforms. The Proposed k-means algorithm will be tested to clarify how it can tackle the major drawback of the classical K-Means algorithm; the random number of clusters is not known in advance and the resolution problem. Experiments will be held on both platforms; Spark and Flink Machine Learning Libraries with large scale data sets that cover all the big data properties

References

[1] O. Marcu *et al.*, "Spark versus Flink : Understanding Performance in Big Data Analytics Frameworks To cite this version : Spark versus Flink : Understanding Performance in Big Data Analytics Frameworks," 2016.

[2] S. Kamburugamuve, P. Wickramasinghe, S. Ekanayake, and G. C. Fox, "Anatomy of machine learning algorithm implementations in MPI, Spark, and Flink," *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 1, pp. 61–73, 2018.

[3] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data K-means clustering using MapReduce," *J. Supercomput.*, vol. 70, no. 3, pp. 1249–1259, 2014.

[4] Q. H. Weizhong Zhao, Huifang Ma, "Parallel K -Means Clustering Based on MapReduce," pp. 674–679, 2009.

[5] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable K-Means++," pp. 622–633, 2012.

[6] H. H. Xiao Cai, Feiping Nie, "Multi-ViewK-Means Clustering on Big Data," in *The Twenty-Third International Joint Conference on Artificial Intelligence*, 2013, pp. 2598–2604.

[7] A. Sinha and P. K. Jana, "A novel K-means based clustering algorithm for big data," *2016 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2016*, pp. 1875–1879, 2016.

[8] L. Ma, L. Gu, B. Li, Y. Ma, and J. Wang, "An Improved K-means Algorithm based on Mapreduce and Grid," vol. 8, no. 1, pp. 189–200, 2015.

[9] P. Arora, D. Virmani, and H. Jindal, "Sorted K-Means Towards the Enhancement of K-Means to Form Stable Clusters," vol. 508, 2017.

[10] A. M. Baswade and P. S. Nalwade, "Selection of Initial Centroids for k-Means Algorithm," *Ijcsmc*, vol. 2, no. 7, pp. 161–164, 2013.

[11] K. Teknomo, "K-Means Clustering Tutorial," *Medicine (Baltimore).*, pp. 1–12, 2006.

[12] "Apache Hadoop," *https://hadoop.apache.org/*, 2019. .

[13] "Apache Spark," *http://spark.apache.org/*, 2019. .

[14] "Apache Flink," *https://flink.apache.org/*, 2019. .