ISSN: 0067-2904

A Parallel Clustering Analysis Based on Hadoop Multi-Node and Apache Mahout

Noor S. Sagheer*[1], Suhad A. Yousif[1]
Department of Computer Science, Al-Nahrain University, Baghdad, Iraq

Received: 3/6/2020 Accepted: 14/9/2020

Abstract

The conventional procedures of clustering algorithms are incapable of overcoming the difficulty of managing and analyzing the rapid growth of generated data from different sources. Using the concept of parallel clustering is one of the robust solutions to this problem. Apache Hadoop architecture is one of the assortment ecosystems that provide the capability to store and process the data in a distributed and parallel fashion. In this paper, a parallel model is designed to process the *k*-means clustering algorithm in the Apache Hadoop ecosystem by connecting three nodes, one is for server (name) nodes and the other two are for clients (data) nodes. The aim is to speed up the time of managing the massive scale of healthcare insurance dataset with the size of 11 GB and also using machine learning algorithms, which are provided by the Mahout Framework. The experimental results depict that the proposed model can efficiently process large datasets. The parallel *k*-means algorithm outperforms the sequential *k*-means algorithm based on the execution time of the algorithm, where the required time to execute a data size of 11 GB is around 1.847 hours using the parallel *k*-means algorithm, while it equals 68.567 hours using the sequential *k*-means algorithm. As a result, we deduce that when the nodes number in the parallel system increases, the computation time of the proposed algorithm decreases.

Keywords: Big Data, Hadoop, Mahout, Predictive Analytics, and Parallel *K*-means.

تحليل التجميع المتوازي بالاعتماد على الهدوب متعدد العقد و اباتشي ماهووت

نور صباح صغير* , سهاد عبد الرحمن يوسف
قسم علوم الحاسوب, جامعة النهرين, بغداد, العراق.

الخلاصة

لإجراءات التقليدية لخوارزميات التجميع غير قادرة على التغلب على صعوبة إدارة وتحليل النمو السريع للبيانات. يعد استخدام مفهوم التجميع المتوازي أحد الحلول القوية لهذه المشكلة. كما وتعد بنية Apache Hadoop أحد الأنظمة البيئية المتنوعة التي توفر القدرة على تخزين البيانات ومعالجتها بطريقة موزعة ومتوازية. في هذا البحث، تم تصميم نموذج متوازي لمعالجة خوارزمية *K*-means في نظام Apache Hadoop البيئي من خلال توصيل ثلاث عقد (خادم واحد وعميلان) لتسريع وقت إدارة النطاق الهائل لمجموعة بيانات التأمين الصحي، وايضا من خلال استخدام خوارزميات التعلم الآلي ، والتي يتم توفيرها بواسطة Mahout Framework. تشير البيانات إلى أن علاج التأمين الصحي ، الذي يتم شراؤه مباشرة من قبل الأفراد والعائلات، عادة ما يكون أرخص وأشمل مع حجم البيانات (11 جيجابايت). توضح النتيجة

*Email: noory_computer@yahoo.com

التجريبية أن النموذج المقترح يمكنه معالجة مجموعات البيانات الكبيرة بكفاءة، وتتفوق خوارزمية متوسطات *k* المتوازية بشكل أفضل من خوارزمية *k* المتسلسلة بناءً على وقت التشغيل ، حيث يكون الوقت المطلوب لتنفيذ (11 جيجابايت) حجم البيانات حوالي (1.847) ساعة باستخدام خوارزمية متوسطات متوازية وباستخدام خوارزمية متوسطات *k* المتسلسلة المطلوبة حوالي (68.567) ساعة. نتيجة لذلك ، نستنتج أنه عندما يزداد عدد العقد في النظام الموازي ، يقل وقت حساب الخوارزمية المقترحة.

1. Introduction

Big data is a combination of large amount, substantial, and multiple formation data created from varied and separated data sources. Researchers and scientists think that big data is one of the most important subjects in computer sciences nowadays [1]. Sensors, sites of social media, hospital annals, and several new foundations are beyond the phenomenon of big Data [2]. A data warehouse cannot deal with the whole dataset because of its vast size [3]. Also, conventional algorithms are incapable of dealing with such enormous amounts of data, so they are not efficient for analysing them [4]. The traditional *k*-means clustering algorithm [5, 6] is not sufficient to manipulate the massive amount of data. Hadoop and Map Reduce tools can be used for dealing with such data [7].

Apache Hadoop Ecosystem is a technology that makes it possible to capture vast amounts of data about each client member and store these data over a considerable time scale [8]. It introduced the concepts of automatic parallelism and distribution. The system of parallel clustering can reduce the large size of data by gathering data that have the same characteristics over the task of data separation to the nodes that are connected in parallel [9]. They are used to speed up the clustering process for large scale datasets. Some researchers [10] suggested a parallel *k*-means clustering algorithm by using the mahout Application Programming Interface (API) to speed up the clustering process for big datasets. They defined a virtual parallel system with 8 GB of RAM and four threads. The results showed that the proposed implementation improved the computation complexity of each iteration from $O(nk)$ for traditional *k*-means to $O(nk/p)$ by separating the complete dataset into (p) subsets.

The contribution in this paper is to deal with big data by proposing a parallel *k*-means clustering algorithm, through designing a real server-client model with a Hadoop multi-node system using Intel® Xeon ® CPU E7-286022.27 GHz x6 processor with llvm pipe (LLVM 7.0,128 bits) graphics, 20 GB of memory, and 133.7 GB, Java1.8.11-JDK, Ubuntu-64-bit-18.04.2 disk. This is assigned as a Name Node. Pentium ® Dual-core CPU E5300@2.60 GHz x2 processors, each with a 101.6 GB disk, 4 GB of memory, and a Ubuntu-64 bit-18.04.2 disk, are allocated as Data Nodes. The nodes are connected using a Transmission Control Protocol (TCP) based protocol. Also, we used Apache Hadoop-2.5.3 (64-bit) version with JDK 1.8.11 to compile the source code.

2. Related Works

The continued growth of the size of datasets has led to traditional clustering methods, but their limits resulted in parallel clustering. In this section, we show some associated clustering algorithms and Hadoop's performance prediction.

In an earlier work [11], the authors introduced an improved method for Parallel *K*-means called IPKMeans that uses the k-d tree for the parallel pre-processing stage. Both IPKMeans and k-d trees are run using MapReduce and Hadoop. IPKMeans is faster than PKMeans because it works with one Map and more reducers in parallel, where the I/O overhead is decreased to 2/3. This was applied to a large dataset and an initial centroid; many reducers caused high speed while keeping very close accuracy. In another article [12], the authors introduced a new system called iiHadoop (incremental iterative) to increase calculations on the small segment of data that is pretentious by vicissitudes instead of all the data. This system recovers the presentation for the Map and reduces responsibilities. In another study [13], researchers depended on a parallel framework for the algorithm of the Partitioning Around Medoids (PAM), which is one of the clustering algorithms. This algorithm is divided into sub-tasks that run in parallel. The outcomes appeared to significantly reduce the computational complexities of the algorithm with an excess in the processing cores number. In another article [14], the authors developed a two-factor clustering-based system to process massive amounts of data. The first factor was an attempt to build a random pattern from specified cluster numbers to be used as a centroid in individual clusters. The second factor was included in the parallel schema of the clustering algorithm for increased speed with high stability and quality results. In a later work [15], the authors used tools such as Hadoop and *k*-means algorithms to analyze and predict data, which is about

26 GB of editions from the sixteenth-century to 2014. They found the age group of authors who wrote most of the books in a certain amount of time. They published more than 3,000 books, which is the number of books published in 2000 by authors who were more than 60 years old. In another study [16], the author proposed a method of robust Hadoop Cloudera, which depended on data tubes to analyse any type and scale of data. This method can be used to detect stocks that are likely to increase in value. The US stocks chosen for predicting profits depend on Yahoo economics data. The researchers used Hadoop to store and process big Datasets and Spark for predicting the stocks. Depending on Yahoo Finance historical data, they found that the Coefficients in a trouble are 0.0817,0.1282,0.0,0.086,0.0,0.0675,0.11,0.0438,0.0046,0.0,0.0, and 0.0, while the regression model intercept value is -0.097938396250894261. They concluded that the linear regression model is not appropriate to predict stocks returning profits from the high dimensionality data because the Average Mean Error equals to 1.95%.

3. HDFS and Mahout framework

Hadoop file system is an essential part of the Hadoop Ecosystem, and it is a file method intended for storing massive datasets through the manifold node of the product hardware [17]. Where the data files are stored on the HDFS as blocks, the version of Hadoop used in this model was 2.x. Therefore, the default size of each block on HDFS was 128 MB.

The Mahout is an exposed machine learning programs group, which is fundamentally a Java archive set [18]. The defined Mahout's core algorithms are counting clustering, classification algorithms [19], design mining and text removal, regression, dimensionality reduction, and batch based cooperative sifting track on the topmost of the Hadoop stage over the Map-Reduce frame [20].

4. Dataset

The used dataset is a group of files in a Comma Separated Values (CSV) file format. The dataset comprises a large amount of data on the healthcare plans available through health insurance. The sizes of these files were different. The purpose of analysing health insurance data was to help the insured persons to choose what is financially appropriate for them in terms of medical and surgical expenses. These files can be loaded from local systems to the Hadoop file system (HDFS). The 'local system' in this model means a personal computer that contains a large dataset. Table-1 shows files with their attributes.

Table 1- File names used in this study and their attributes

File name	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5
Benefits cost-sharing	Benefit Name	Business Year	CoinsInnTier1	CoinsInnTier2	Coins Out of Net
Business Rules	Business Year	State Code	Issuer ID	Source Name	Dental Only Plan
Crosswalk 2015	State Dental Plan	Plan ID_2014	IssuerID_2014	Child Adult Only_2014	Dental Only Plan
Crosswalk 2016	State Dental Plan	Plan ID_2015	IssuerID_2015	Child Adult Only_2015	Dental Only Plan
Network	Business Year	State Code	Issuer ID	Source Name	Version Number
Plan attributes	Benefit Package ID	Business Year	CSR Variation Type	Child Only Offering	Child Only Plan ID
Rate_PUF	Business Year	State Code	Issuer ID	Source Name	Individual Tobacco Rate
Service Area	Business Year	State Code	Issuer ID	Service Area Name	Source Name

5. Proposed Model and Work Flow

The large quantity of data requires high cost for computational operations, while when applying the clustering techniques on the scale size of data, high quality was achieved. The clustering is helpful for analyzing the data. As a step of preprocessing for some learning tasks, using clustering can improve

the accuracy of prediction during obtaining much information about data [21]. Figure- 1 explains the whole system for analyzing big data.

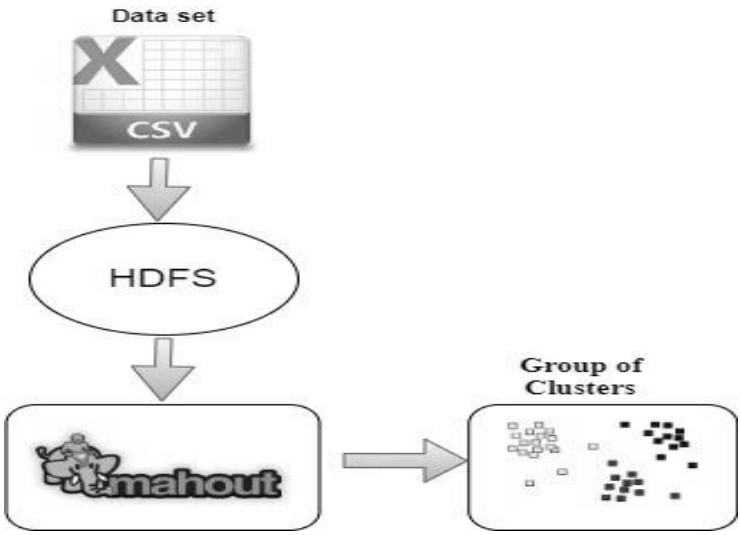


Figure 1- General workflow of the proposed model for analyzing big data

The explanation of the proposed model stages and how an extensive dataset is processed are illustrated in Figure- 2.

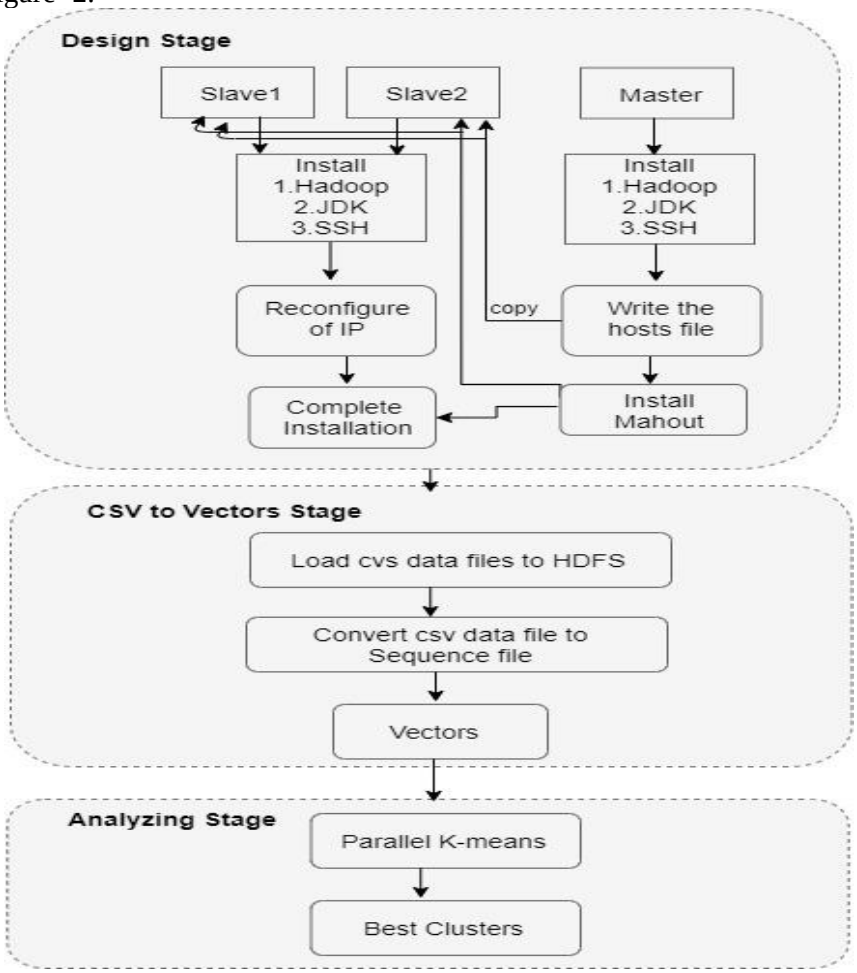


Figure 2- General workflow of the proposed model for analyzing big data

5.1 Design Stage

The design stage is a crucial stage in the proposed model, which includes the hardware components represented by the master node and two slave nodes and their configurations. The software requirements were set as installing Hadoop single node clustering, Hadoop multi-node clustering, and Mahout and Ubuntu installation. Finally, the connection between the master and slave nodes is performed.

5.2 CSV to Vectors Stage

Before processing the data in Mahout, the data must be uploaded to HDFS. The following steps were applied:

1. Start running the Hadoop server in the Master node. The overview of the master node "noor-Virtual Machine-9000" shows that it is active now. Other details of Hadoop windows are shown in Figure- 3.

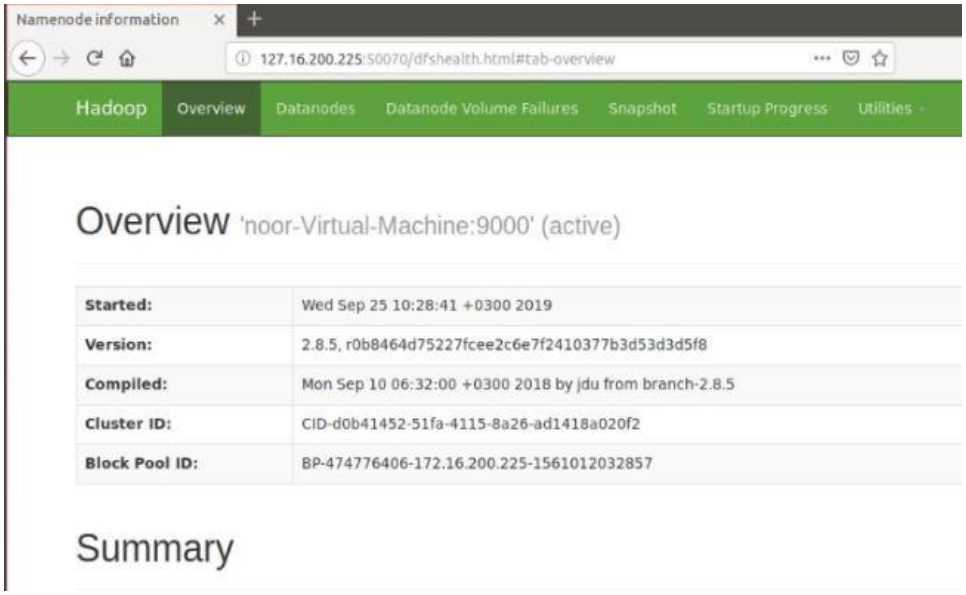


Figure 3- Snapshot of hadoop's window.

2. Make directories to store the csv-files in the HDFS file system.
3. Copy the data file from the Linux file system (Ubuntu) to the HDFS file system by using the copy command ***bin/$hdfs dfs –put /…path to file in local system/ /…path*** *to Hadoop file system*.

Figure- 4 shows an example of how data is saved as blocks in HDFS, where a Rate_BUF.csv is one of the healthcare dataset files with the size of 842 MB; therefore, the number of blocks to store in HDFS is seven, as described below:

842 MB ÷ 128MB = 6.578

The division result shows that six out of all the seven blocks stored 128 MB, while the last block stored the remaining (74 MB).

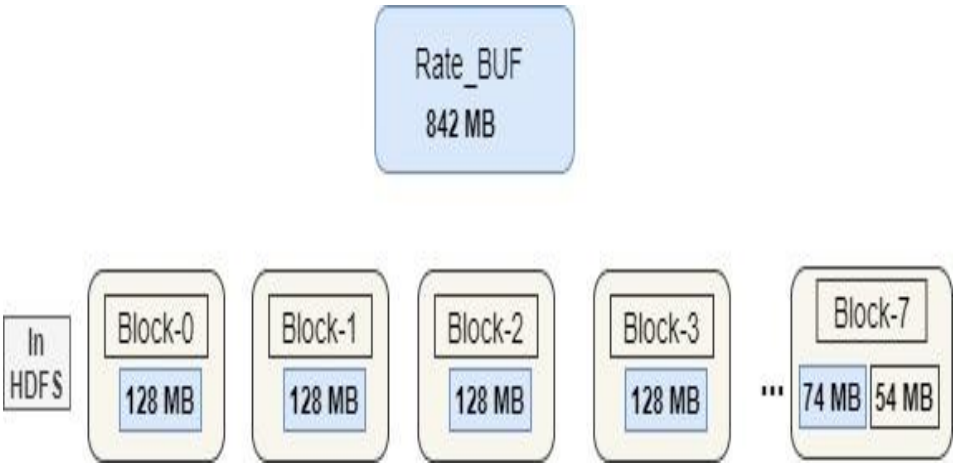


Figure 4- Blocks of data in HDFS.

Figure- 5 illustrates two images of the Rate_BUF file, one for the file properties and the second for the file with its blocks in HDFS.

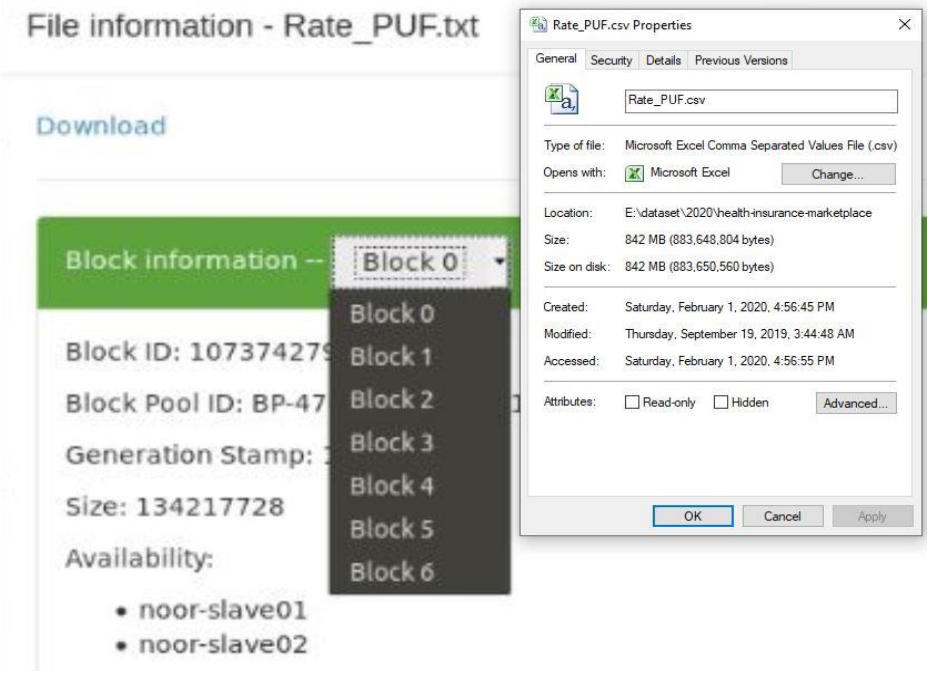


Figure -5 Screenshot of rate-PUF file and HDFS blocks.

The process of writing the generated blocks into HDFS multi-nodes, where the size of the dataset is 11 GB stored into more than one file of different sizes, is shown in Figure- 6. According to the size of each file, the number of generated blocks will be 107 blocks (from Block0 to Block106), and it will be saved randomly in the HDFS of the two defined data nodes, as shown in the following steps:

A. The client node sends a request to the name node (noor-Virtual-Machine). The name node will provide the map where data is and where data should be saved in the HDFS. The name node has saved only the metadata of all data nodes.

B. The name node checks the ready data nodes and returns the IP addresses of the available data nodes.

C. The client node will then check if the specified data node is ready to write blocks randomly on the specified IP addresses of the data nodes (noor-slave01, noor-slave02). The replication factor is one, so only one copy of the data will be saved on the ready data nodes. This is due to the proposed model in which one rack with one name node and two data nodes is introduced.

D. Finally, an acknowledgment message is sent to the client node by the data node, which tells that the writing process is completed, and in turn, an acknowledgment message is sent to the name node by the client node, which tells that the blocks are written successfully. Finally, the metadata of the name node will be updated.

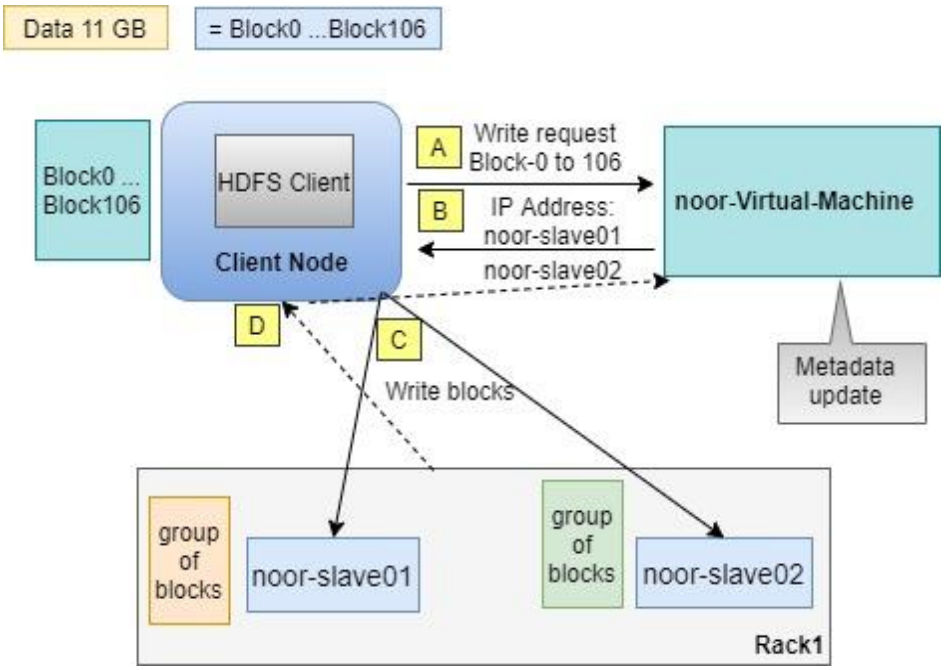


Figure 6- Diagram of writing 11-GB data files to HDFS multi-node

The process of reading the saved blocks from the HDFS multi-node for processing steps is presented in Figure- 7. The number of blocks that will be read is 107 blocks (from Block0 to Block106) that are saved randomly in the HDFS of the two defined data nodes. The following steps are applied for the mechanism of reading blocks:

A. The client node sends a read request to the name node (noor-Virtual-Machine). The name node will provide the map of where data is written in the HDFS.

B. The name node checks the data nodes that have the required blocks and returns the IP addresses of the available data nodes.

C. The client node will then check if the specified data node is ready on the data nodes' (noor-slave01, noor-slave02) for the specified IP addresses.

D. Finally, the data node sends the required blocks to the client node and then sends an acknowledgment message to the client node, telling that the reading process is completed, and in turn, the client node sends an acknowledgment message to the name node, telling that the blocks are read successfully. After that, the metadata of the name node will be updated.

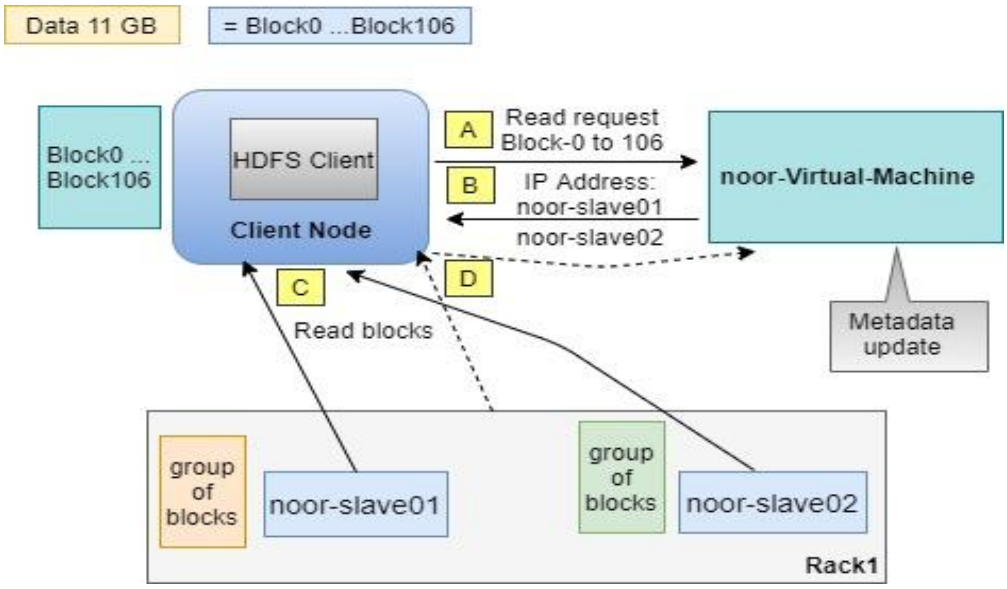


Figure 7- Diagram of reading 11-GB data files from HDFS multi-node

The job of Hadoop is to change the data from CSV format to the Hadoop sequence-file format. The Hadoop sequence-file is a structure of flat-file data containing (key, value) couples in binary. The mapper of Hadoop service is the input reader for parsing input key and value. As the default format of the Hadoop input reader is a text input anywhere, each text line signifies a record. However, this is not appropriate for the CSV format because technical provision requests to cross many appearances. Thus, a reader and partitioner of the custom input record are required for the proposed process. The reader of the input record gathers text from the input file until it reaches the identified record sign end. The mapper needs 'key, value', where the value is a text resulting from the reader of an input record, while the key is adjusted to the location in the record's file in order to extract and pass them to the reducer. Finally, the reducer obtains these (key, value) couples and writes them into a Hadoop sequence file format to be processed using Mahout Commands. The following is the mahout command to convert the data into the sequence file:

$mahout seqdirectory -i /path to Dataset file in hdfs/ -o /path to output file in hdfs/ -ow

5.3 Analysis Stage (parallel *k*-Means clustering algorithm)

When the data are loaded to the Hadoop file system, one of the denominations below must be selected to improve the speed and the scalability of big data analysis:

1. Clustering algorithm of the single machine (Shared Memory System): it is operated in one computer and used only its sources, and includes the sample-based and dimension-reduction techniques.
2. Clustering algorithm of the multi-machine (Distributed system): it is operated in multi (computer/server) and can run in one computer and access other sources. It includes the parallel clustering and Map-Reduce based clustering. This technique deals with a large dataset.

In this paper, a multi-machine is used for implementing the clustering algorithm. The *k*-means algorithm can be implemented in two modes:

Serial mode: When the *k*-means algorithm is implemented in the serial mode, the initial value of *k* centers of clusters is chosen randomly; after that, the data files are loaded from HDFS. Then the algorithm computes the distance between all the data points and the initial centers of clusters, and it allocates the data point to the cluster that has the nearest center to it. When the first round finishes re-computing the new centers of the clusters, the algorithm repeats the process until it reaches the number of iteration specified by the user, or to no change in the value of clusters centers. Finally, all the generated clusters are written to HDFS. Algorithm- 1 below lists the work of serial *k*-means clustering.

Algorithm 1: serial *k*-means clustering

Input:
Sequence files Sequence.vec; initial centers K; and Threshold X

Output:
Group of clusters

Procedure:

Step1: def dis←[]; V←[];
Step2: Try (BufferedReader seq = new BufferedReader(new File Reader (path to sequence.vec)))
\\ readfile from hdfs to seq buffer
Step3: String line;
Step4: While seq.readLine() != null
Step5: line ← seq.readLine();
Step6: For i=0 →length(line)
Step7:V[]←line.split(//t); \\ split each line by tab
Step8: End For
Step9: End While
Step10: catch(System.Exception ex); \\ end try
Step11: Repeat
Step12: For j=0 →K
Step13: For $m_i \in$ V[]
Step14: dis[]= $\|m_i - K[j]\|$; \\ compute the distance between current vector and K center,$1 \leq i \leq V$
Step15: End For; \\ end for i

Step16: End For; \\ end for j
Step17: Assign vectors to the nearest cluster;
Step18: Recalculate new centers K;
Step19: Until X || K no change; \\ repeat until satisfying threshold or stable K reached
End

Parallel mode: There are two categories for the parallel algorithm, which are the control-parallel and data-parallel. In data-parallel, the extensive dataset is divided into small datasets, and parallel computing is conducted on each set of data. In the control parallel, diverse computing is conducted with each small dataset [10]. In this paper, data paralleling is used for implementing the parallel k-means algorithm because it has good scalability and it is appropriate for processing large datasets. First, the algorithm splits the large dataset into the small datasets after reading them from HDFS. For each dataset, it is processed in a single data node which is used to compute the distances between the clustering center and each data point. When all data points are allocated, the new clustering centers are computed in the master (name node). Then new centroids are re-assigned to the data nodes. This algorithm finishes when the specified number of repetitions has been met or no change in the centroid value. Furthermore, the number of the data node matches the subset's number, and the master node communicates to the data nodes. This communication plays a significant part in parallel mode. Algorithm- 2 below represents the steps of the parallel *k*-means clustering algorithm implemented using Java code.

Algorithm 2: parallel *k*-means clustering

Input: sequence files sequence.vec; Canopy centers K; and Threshold X.
Output: K clustering centers with dictionary file (ID, term).
Procedure:

tep1 : Let Nd: number of nodes; Let V a list; Let Sb: the subset of V; Vc: vector center, nc: number of elements in each cluster; nd: child of nodes; NC= new center for each node;
tep2 : Try (BufferedReader seq = new BufferedReader(new File Reader (path to sequence.vec))) \\ readfile from HDFS to seq. buffer
tep3 : String line;
tep4 : While seq.readLine() != null
tep5 : line ← seq.readLine();
tep6 : For i=0 →length(line)
tep7 :V[]←line.split(//t); \\ split each line by tab
tep8 : End For
tep9 : End While
ep10 : catch(System.Exception ex); \\ end try
ep11 : :Repeat
ep12 :For i=1 →Nd do \\ where Nd=2
ep13 : Assign Sb/Nd \\ assign sub data to each node
ep14 : Assign K/Nd \\ assign canopy center to each node
ep15 : Assign Vc/Nd \\ assign vector center to each node
ep16 : End For
ep17 :For each nd ∈ Nd do \\ loop to slave node
ep18 : For j=1→ Sb do
ep19 : For i=1→ K do
ep20 : $Dist_{sb}[]$ =distance(Sb_j, K_i) \\compute the distance between vectors and cluster center
ep21 :End for; \\ for i
ep22 :End for; \\ for j
ep23 :For all instances in Sb assigned to nearset cluster f, 0>f>K do
ep24 :End for; \\ for all instances
ep25 :End for; \\ for of loop to slave node
ep26 : Recalculate the new centers of clusters by gathering vectors in each cluster and dividing it

on numbers
ep27 **:** Until X|| no change K**;** \\ repeat until satisfying the condition
End

Finally, the cluster dumper is a utility provided by Mahout and used to read the contents of the resulted files of the *k*-means algorithm and store it as a text file. The following command shows how we can read the outcomes of the algorithm by using the dictionary file:

$mahout clusterdump -i /path to/clusters--final –dt sequencefile -d /path to dictionary file –of TEXT –p /path to clusterdpoint –o /path to the output file in the local system –b –n*

Where –i means input file path, -d is the dictionary file path, -p is the points of cluster, -o is the output file path, -b is the substring to print, -n is the number of words in top terms.

6. Results and Discussion

The experiments are based on the initial number of clusters and the run time for both parallel and sequential *k*-means clustering. The defaulting *k*-means algorithm is run to find four clusters. Throughout our experiments, we set bounds to several active nodes. For sequential execution, the whole algorithm is tracked as a single processor on a single node. For parallel execution, the number of the active processor is restricted by the number of nodes.

Figure- 8 shows the execution time when loading the data from the local system to the Hadoop system for both the sequential and parallel implementations with different data sizes. In a single node, the time spent to load 5 GB was 11.9 minutes, while the three-nodes required 0.98 minutes. Moreover, when loading 11 GB in multi-node, the time spent was 2.02 minutes, whereas 27.3 minutes were spent at the single node. The time was increased in both cases because of the increases in the size of the dataset, but time at the single node was increased more than ten times as much as that at the multi-node.

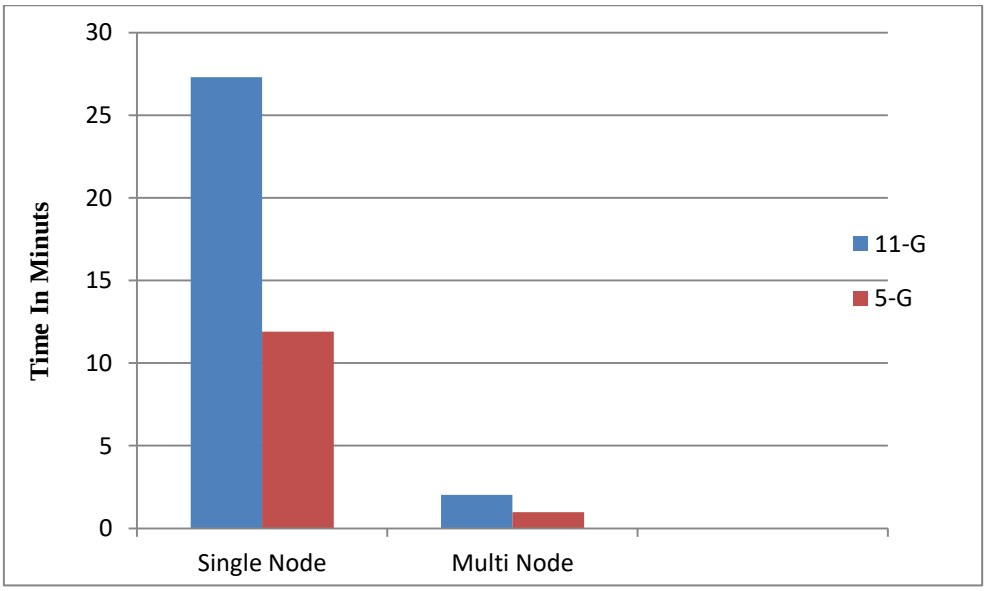


Figure 8- Comparison of time for single node and multi-node

Table- 2 and Figure- 9 show the total time for clustering the datasets that have 11GB with all details for both sequential and parallel implementations.

Table 2- Time in minutes to analyze data (0.011TB) on a Hadoop Cluster for both Sequential (one node) and Parallel (three nodes) Implementations.

EM	LT in Min.	CSV to SEQT in Min.	VT in Min.	KT in Min.	TT in Min.
Sequential	27.3	73.401	151.66	4114	4366.361
Parallel	2.75	7.97	18.26	110.8	139.78

EM: Execution Model.
LT: The time of load of datasets form a local system to Hadoop file system.
CSV to SEQT: The time for converting the .CSV files to Sequence file.

VT: The time for converting sequence-file to Vectors.
KT: Time to run the *K*-means algorithm and TT: Total Time.

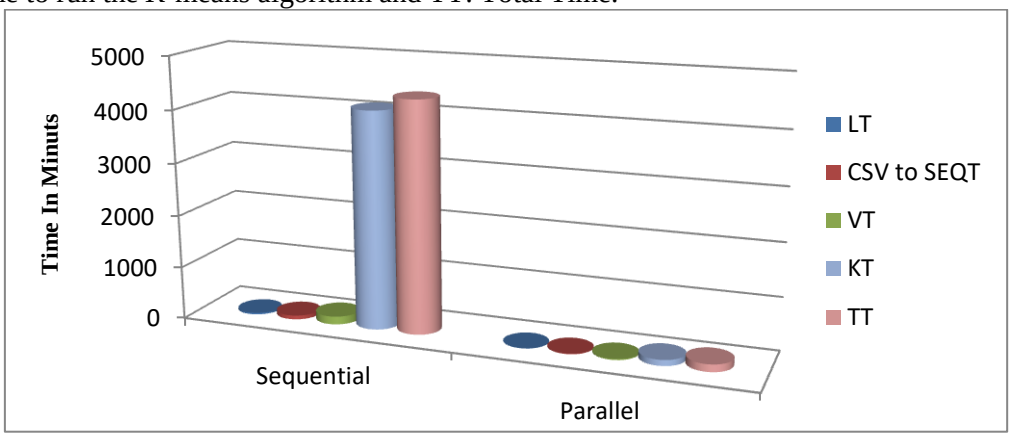
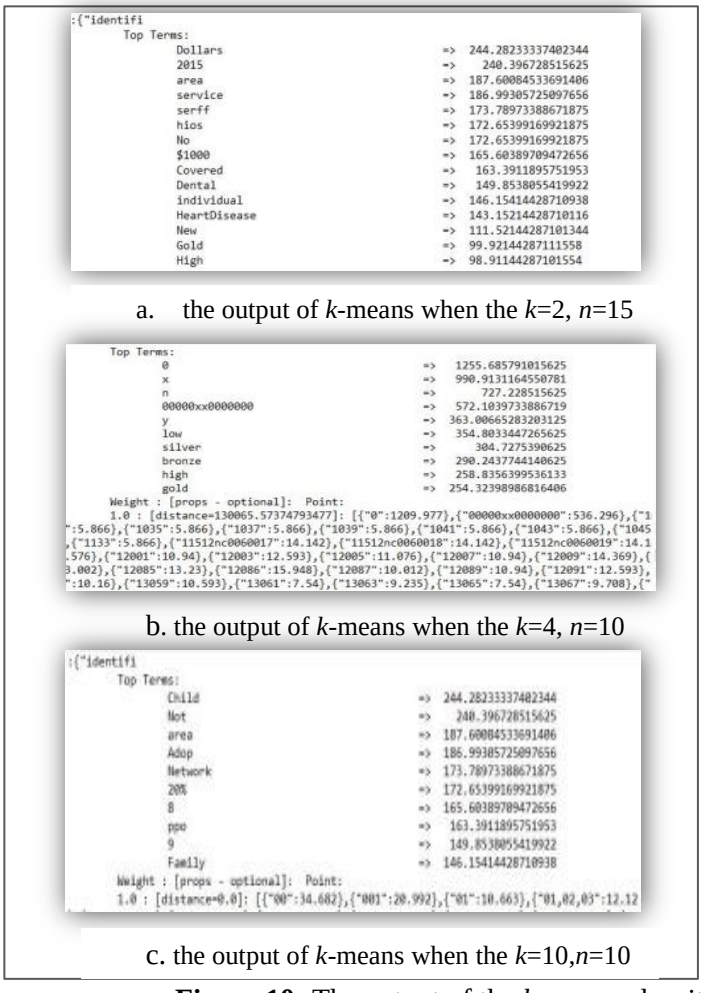


Figure 9- Total time required for clustering 0.011TB at single and multi-nodes.

There are several attempts to choose the best *k* (number of clusters) for analysing 11 GB of data. In this paper, we chose the best *k*, depending on the resulting clusters. This means that when we select a number for *k*, the cluster outputs should be accurate and can be analysed. Hence, when the number of clusters equals two, as in Figure- 10.b, there are some undesired words, such as *y*, 0, and *x*. When *k* = 7, good topic terms were discovered (Figure- 10.a). However, when *k* = 10 and higher, undesired words arose (Figure- 10.c), and the clusters were inaccurate. Whereas, *n* means the number of words that appear in the text file.



a. the output of *k*-means when the *k*=2, *n*=15

b. the output of *k*-means when the *k*=4, *n*=10

c. the output of *k*-means when the *k*=10,*n*=10

Figure 10- The output of the *k*-means algorithm

Figure- 11 shows the performance of the K-means algorithm for the clustering of 11 GB of data. In both serial and parallel modes, the time required was increased when the number of clusters was increased. This is because the increased distance computations between the centers of clusters and their elements. When we selected k = 2, the time spent to run the algorithm mentioned above in the parallel mode was 1.7 hours, while in the sequential mode it was 65.367 hours. Also, when we selected k = 4, the time required in the sequential mode was 68.567 hours, while in the parallel mode it was 1.846 hours. In the same way, when we selected k = 10, the time required in the parallel mode was 4.556 hours, while in the sequential mode it was 81.683 hours.

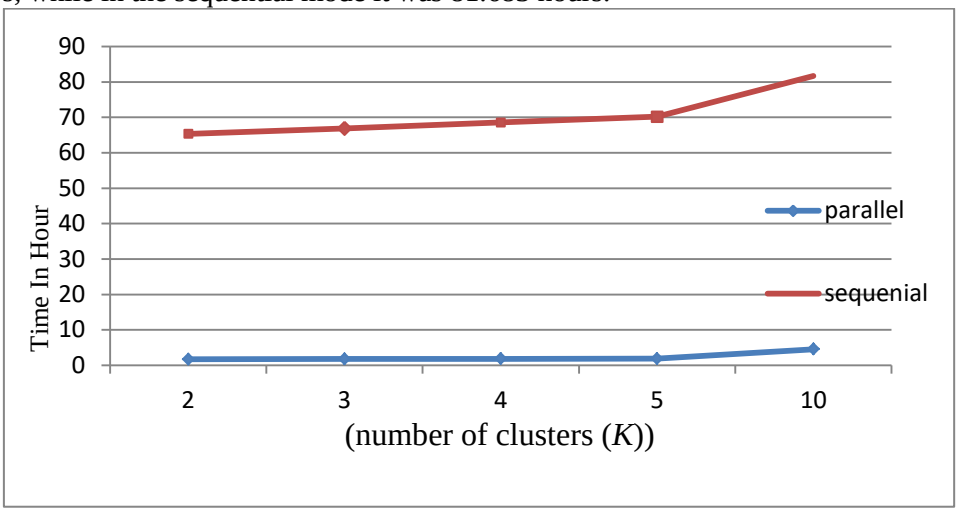


Figure 11- Performance of parallel and sequential *k*-means algorithm with changing clusters number

Table- 3 shows the seven clusters resulted after applying the parallel *k*-means clustering algorithm with some of their contents (some attributes are selected manually from the files mentioned in Table-1). The one cluster contains several features as those in the other clusters, but different that is distinguishing each cluster. These features can be adopted by individuals and small businesses to choose a suitable health insurance offer. Families or individuals with excellent income are classified within the cluster1, while those with limited income are classified within the cluster5 or cluster7. For example, cluster1 provides benefits such those for dental accidents, dialysis, denture adjustment, and disease management programs. Also, it is appropriate for people who do not prefer smoking and have no conditions for their ages, as well as, for example, when the metal used for dental treatment is gold. Furthermore, the participants can have insurance as a primary subscriber with one dependent (adult or child). While, cluster3 provides benefits such as acupuncture, root canal therapy for children, and blood test services. Also, it is appropriate for persons who are tobacco smokers, their ages range between 20-40 years, and when the metal used for the dental treatment is silver. Furthermore, the participants can have insurance as a couple and with one dependent (only one child).

Table 3- The seven clusters with some of their contents.

Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7
* Dental Accidents * Dialysis *Denture Adjustment * disease manage programs	*Orthodontia Child *major child *Bridges Child *Hospice Services	*Acupuncture *Root Canal Therapy Child *Blood and Blood Services	*basic dental care adult *major adult *Home Health Care Services - Visits 21-60	*dental checkup for children *basic dental care child	*weight loss programs *Orthodontia Child *Cancer Drug Administration	*Routine dental service (Adult) *Denture repair – Child *Adult Frames or Lenses

*99%	*65%	45%	40%	35%	66%	20%
*No preference	* Non-tobacco user	Tobacco user	No tobacco	Tobacco and no tobacco	* preference	*unknown
*Shop	*Individual	*Shop (Small group)	*Small group	*Shop	*Individual	Small group
*family option	*50,58,59	*20,22, 30,37	*60-64	*0-20	*65 and over	*40,48,49
*HIOS	*SERFF	*HIOS	*HIOS	*SERFF	*OPM	*SERFF
*Gold	*High	*Platinum	*Silver	*Catastrophic	*Low	*Bronze
*Value service Area	* Geisinger Health Plan	*Blue preferred PPO	*Dental managed care	*Medica applance	*Bluecross and blue shield	*Service area 1
*Best live DM	*Aeches Horizon Network	*Altius network	*Alliance	*Access care	*Dentegra dental PPO	*Ameritas PPO
*More than three	*Age on insurance date	*Age on birthday	*More than 3	*3 or more	*Age on January	*two years
*Adult and child	* Child only	*Child only	*Non	*Adult and child	*Adult and child	*Adult and child
*yes	*No	*yes	*No	*yes	*yes	*yes
*Primary subscriber and one dependent	*Couple and two dependent	* couple and one dependent	* couple	* couple and more than two dependent	* primary subscriber and more than two dependent	* primary subscriber and two dependent

7. Conclusions

During the implementation of the proposed model that is based on big data analysis technology, several deductions are adopted based on the produced results. The next points describe the substantial ones:

1. The performance of parallel k-mean through Hadoop multi-node is presented regarding the time of implementation and the nodes number. So, when the nodes number rises, the time execution decreases.
2. The algorithm of parallel k-means performs in a good and professional manner, and the consequences are based entirely on the size of Hadoop clusters.
3. The time to implement the k-mean in the multi-node (parallel) is less than that in a single node (serial).

References

1. A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, **2014.** "Big data clustering: a review," in International Conference on Computational Science and Its Applications, 2014, pp. 707-720: Springer.
2. Y. Lamari and S. C. J. J. o. B. D. Slaoui, **2017**. "Clustering categorical data based on the relational analysis approach and MapReduce," **4**(1): 28, 2017.
3. A. D. Barrachina and A. J. J. o. B. D. O'Driscoll, **2014.** "A big data methodology for categorizing technical support requests using Hadoop and Mahout," **1**(1): 1, 2014.
4. U. Kazemi, **2017.**"Clustering methods in Big data," *Journal of Embedded Systems and Processing,* **2**(3): 1-5, 2017.

5. A. H. Ali, L. A. Al Ani, and L. A. George, **2014.** "Clouds Height Classification Using Texture Analysis of Meteosat Images," *Baghdad Science Journal*, **11**(2): 652-659, 2014.

6. H. H. MAALA and S. A. YOUSIF, **2019**. "Cluster Trace Analysis For Performance Enhancement In Cloud Computing Environments," *Journal of Theoretical and Applied Information Technology*, **97**(7), 2019.

7. S. A. Babu, N. Vijay, and R. Gadde, **2017.** "An Overview of Big Data Challenges, Tools and Techniques," 2017.

8. Belle, R. Thiagarajan, S. Soroushmehr, F. Navidi, D. A. Beard, and K. Najarian, **2015.** "Big data analytics in healthcare," BioMed research international, vol. 2015, 2015.

9. S. S. Bandyopadhyay, A. K. Halder, P. Chatterjee, M. Nasipuri, and S. Basu, **2017.** "HdK-means : Hadoop based parallel K-means clustering for big data," in Calcutta Conference (CALCON), 2017 IEEE, 2017, pp. 452-456: IEEE.

10. X. Daoping, Z. Alin, and L. Yubo, **2016.** "A parallel Clustering algorithm implementation based on Apache Mahout," in 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 2016, pp. 790-795: IEEE.

11. S. Jin, Y. Cui, and C. Yu, **2016.** "A New Parallelization Method for K-means ," arXiv preprint arXiv:1608.06347, 2016.

12. A. G. B. Saadon and H. M. Mokhtar, "iiHadoop: an asynchronous distributed framework for incremental, iterative computations," *Journal of Big Data*, **4**(1): 24, 2017.

13. H. Mushtaq et al., **2018.** "A Parallel Architecture for the Partitioning around Medoids (PAM) Algorithm for Scalable Multi-Core Processor Implementation with Applications in Healthcare," *Sensors*, **18**(12): 4129, 2018.

14. R. Benaissa, F. Benhammadi, O. Boussaid, and A. Mokhtari, **2018.** "Clustering Approach for Data Lake Based on Medoid's Ranking Strategy," in International Conference on Computer Science and its Applications, 2018, pp. 250-260: Springer.

15. R. Solanki, S. H. Ravilla, and D. Bein, **2019.** "Study of Distributed Framework Hadoop and Overview of Machine Learning using Apache Mahout," in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 0252-0257: IEEE.

16. Z. Peng, **2019.** "Stocks Analysis and Prediction Using Big Data Analytics," in 2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), 2019, pp. 309-312: IEEE.

17. J. Archenaa and E. M. Anita, **2015.** "A survey of big data analytics in healthcare and government," *Procedia Computer Science*, **50**: 408-413, 2015.

18. A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, **2018.** "Big Data technologies: A survey," *Journal of King Saud University-Computer and Information Sciences*, **30**(4): 431-448, 2018.

19. S. A. Yousif, Z. N. Sultani, and V. W. Samawi, **2019.** "Utilizing Arabic WordNet Relations in Arabic Text Classification: New Feature Selection Methods," *IAENG International Journal of Computer Science*, **46**(4), 2019.

20. D. P. Acharjya and K. Ahmed, **2016.** "A survey on big data analytics: challenges, open research issues, and tools," *International Journal of Advanced Computer Science and Applications*, **7**((2): 511-518, 2016.

21. S. Trivedi, Z. A. Pardos, and N. T. Heffernan, **2015.** "The utility of clustering in prediction tasks," arXiv preprint arXiv:1509.06163, 2015.