Taylor & Francis
Taylor & Francis Group

Journal of Discrete Mathematical Sciences and Cryptography

ISSN: 0972-0529 (Print) 2169-0065 (Online) Journal homepage: https://www.tandfonline.com/loi/tdmc20

Performance evaluation of K-means clustering on Hadoop infrastructure

Satvik Vats & B. B. Sagar

To cite this article: Satvik Vats & B. B. Sagar (2019) Performance evaluation of K-means clustering on Hadoop infrastructure, Journal of Discrete Mathematical Sciences and Cryptography, 22:8, 1349-1363, DOI: 10.1080/09720529.2019.1692444

To link to this article: https://doi.org/10.1080/09720529.2019.1692444

Published online: 13 Jan 2020.

Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 22 (2019), No. 8, pp. 1349–1363
DOI : 10.1080/09720529.2019.1692444

Taylor & Francis
Taylor & Francis Group

Performance evaluation of K-means clustering on Hadoop infrastructure

Satvik Vats *
B. B. Sagar †
Department of Computer Science and Engineering
Birla Institute of Technology
Mesra 835215
Ranchi
Jharkhand
India

Abstract

Today we are living with the extensive volume of information which is developing at a very fast pace. Clustering is the process of group similar kinds of information. The serial k-means clustering method takes a large amount of computational time when it is applied to enormous data sets which are large in size of tera and petabyte. In this paper, we evaluated the performance of the K-means algorithm in different ways like K-mean simple (using java codes on MapReduce), K-means (using java codes on MapReduce) with IEC (Initial Equidistant Centres), K-mean on Mahout (using Machine learning library) and K-mean on Spark (Machine learning library) on static IP address data sets over the MapReduce framework. In addition to this we also compare the K-mean simple and K-mean (IEC) on different iteration level. Outcome shows on the better performance on the basis of time on different infrastructure and it also defines the behaviour of K-means algorithms on the basis of centroids and different iteration levels.

Subject Classification: *Big Data analytics*

Keywords: *MapReduce, Hadoop, HDFS, K-means, Clustering, Big Data, Spark, Mahout.*

1. Introduction

A cluster is a collection of similar types of data, which have similarities with other data objects within the same cluster. For Detecting an unusual

*E-mail: satvik.vats@gmail.com (Corresponding Author)
†E-mail: bbsagar@bitmesra.ac.in

object like outliers, clustering is suitable algorithm [13]. K-means clustering is the partitioning method, which divides all n-objects into K-clusters in such a manner that it provides high intra-cluster similarity and low inter-cluster similarity.

In the present scenario manufacturing personalization is directly related to the industrial big data. It comes with multi-dimension model having strong correlation and high throughput which helps to improve the value chain and personalization system [16]. A tool assortment and optimization procedure for the big data analytics was introduced in order to effective use of production-process-data (PPD) and to improve the accuracy of tool choice [14].

In the present paper, we have used the Hadoop framework that provides a platform that deals with the structured, semi-structured, and unstructured data sets. Hadoop has two main components, one is HDFS (Hadoop Distributed File System) and the other is MapReduce. HDFS is designed to handle a large amount of data set with strong reliability. MapReduce is the programming model of the Hadoop framework. It is designed in such a manner that one can process a large volume of data sets in parallel. In other words, we can say it provides a facility to run the application into a distributed environment. This paper is organized as follows. Section 2 presents a descriptive study on K-means, MapReduce as a programming model. Section 3 presents the methodology to run the K-means on MapReduce. Section 4 presents the experimental results for performance evaluation on 4 ways of K-means algorithm and provides the comparison between the different iteration level on the dataset. At last, we conclude in section 5 with possible future research direction.

2. Descriptive Study

Why we used the K-Mean clustering algorithm? The answer is K-mean has a wide area and it can cluster millions of data based on occurrence, time of generation, iteration and on similarities. The K-means clustering speaks to an iterative MapReduce calculation and utilized it to group up to 40 million information points needing around 250 MapReduce iterations. Let' begin with an introduction to today's world of digital data and try to figure out how the K-Means fits in this real-world scenario. With the fast improvement of the Internet, tremendous volumes of records are generated over the web. In some of the cases, there was a maximum possibility of having a similar type of data. For example: from a region or place having the maximum ping on a website etc. This creates a scope of research on web

mining, which focuses on a scalable system, which is an application to mass documents [2]. Storage and processing of mass reports information in a dispersed framework is an alternative system [6]. In circulated processing, an issue is separated into numerous undertakings, each of which is solved by one commodity hardware also called as Personal computer (PC). On the other hand, numerous issues, for example, task scheduling; fault tolerance and inter-machine communication is exceptionally dubious for software engineers with little involvement with parallel and appropriated framework. MapReduce [4] is a system in which software engineers just need to indicate Map and Reduce capacities to make a huge task parallelize and execute on an expansive cluster of merchandise machines. The above two information analysis utilizing Hadoop and CGL MapReduce and compare the outcomes. Outcomes confirm the following perceptions [5].

- Most scientific information examinations, which has some type of Single Program Multiple Data (SPMD) calculation can benefit from the MapReduce strategy to accomplish speedup and adaptability.
- As the amount of information and the amount of calculation builds, the overhead affected by a specific runtime reduces.
- Even firmly coupled applications can profit by the MapReduce system if the appropriate size of information and an effective runtime are utilized.

They demonstrate that a few elements, for example, the need of getting binary data, the utilization of diverse programming languages, and the utilization of iterative algorithm, displayed by logical applications may confine the current's appropriateness of MapReduce executions specifically to the scientific information preparing assignments.

Map-Join-Reduce, a system that extends and improves MapReduce runtime structure to beneficially handle complex data analysis tasks on a large cluster. The principal job applies to sift rationale to all the data sets in parallel, joins the qualified tuples, and pushes the join results to the reducers for partial aggregation. The second employment solidifies all partial aggregation results and creates the last reply. The benefit of their methodology is that they join various information sets in one go and thus avoid frequent check guiding and rearranging of intermediate results, a major performance bottleneck in the greater part of the current MapReduce based frameworks [8]. K-means is a standout amongst the most renowned clustering algorithm, the traditional technique of

data mining has low precision. It can be improved by fuzzy clustering method [7].

Its effortlessness and velocity permit it to keep running on vast information sets. Nonetheless, it likewise has a few disadvantages. In the first place, this calculation is unstable and sensitive to exceptions. Second, its execution will be wasteful when managing expansive information sets. A system is proposed to take care of those issues, which utilizes an outfit learning technique bagging to overcome the insecurity and affectability to outlier while utilizing a distributed processing structure, MapReduce, to take care of the wastefulness issue in a grouping on huge information sets [15]. As the quantity of nodes expands the execution time diminishes, additionally a fascinating percentage of cases have been found at the time of experiment and recorded the different execution change and drawn diverse execution graphs [10]. A Sample-based Hierarchical Adaptive K-means (SHAKM) clustering procedure is used for large scale feature recovery. To handle extensive databases proficiently, SHAKM utilizes a multilevel arbitrary examining technique. Furthermore, SHAKM uses the adaptive K-means clustering algorithm to focus the right number of groups and to develop an unequal cluster tree. SHAKM utilizes the quick labelling scheme to relegate every pattern in the dataset to the nearest cluster. To evaluate the proposed technique, a few datasets are utilized to illustrate its effect. The outcomes demonstrate that SHAKM is quick and compelling on vast datasets. Moreover, the outcomes exhibit that the proposed system can be utilized effectively and successfully for a task on substance-based video copy detection [11]. The preparation of extensive scale information utilizing the K-means clustering algorithm and propose a novel handling model in MapReduce to dispose of the iteration dependence and acquire high performance. The iteration reliance of K-means is the greatest bottleneck when we handle huge information in MapReduce. Secondly, a novel technique to improve the K-means clustering procedure using MapReduce wipes out the reliance on iteration and lessens the calculation expense of algorithm. The implementation characterizes the mapper and reducer jobs and obliges no changes to the MapReduce system. Thirdly, they propose two sample consolidating methodologies for their handling model and conduct extensive tests to consider the impact of different parameters utilizing two genuine datasets and one synthetic dataset. The outcomes demonstrate that their proposed routines are proficient and adaptable [3].

A parallel K-mean clustering algorithm depends on MapReduce. This MapReduce is a fundamental and successful parallel programming

strategy. The test outcomes demonstrate that the proposed estimation can scale well and beneficially prepare huge datasets on commodity equipment. They assess the execution of the proposed calculation as for speedup, scaleup, and size up [15]. To evaluate the speedup in two ways, first they kept the dataset reliable and expand the number of PCs in the framework and second, they have performed the speedup appraisal on datasets with distinctive sizes and frameworks. The number of PCs changed from 1 to 4. Scaleup assesses the calculation's capacity to grow both the framework and the dataset size. Scaleup is described as the limit of an m-times greater framework to perform m-times greater employment in the same run-time as the first framework. The size up analysis holds the number of PCs in the framework steady and builds up the span of datasets by the element m. Size up measures the amount of time it takes on a given framework when the dataset size is m-times greater than the first dataset.

Parallel K-mean clustering algorithm which employs an SPMD methodology based on an information-sharing model. Sharing information between a master and the simultaneously made process is done asynchronously. In this manner, the execution can be parallel and fault-tolerant. The experimental results show an extensive speedup rate of the proposed parallel k-means clustering technique, contrasted with the serial k-means approach [9]. The experimental results uncover that the parallel system impressively speedups the calculation time, particularly when tried with multi-core processors. The estimated plot likewise delivers adequate results in a short period of running time.

3. Methodology

3.1 *Map-Reduce Function*

In this paper we implement K-Means (with Initial Equidistant Centers) on MapReduce using Hadoop platform and run this algorithm on internet surfing data of static IP address. When the jar file fetches the input file, it splits into equal size and broadcast to all the mappers. The input to the Map function is given into the form of <key, value> pair, where the key is defined as the cluster center and value is for an object (vector) which uses to be a cluster. According to the MapReduce discipline, it uses two files, one that creates clusters with their centroid and others that houses the objects which are to be clustered [1]. Algorithms for K-Means mapper function are as follows:

Initialization - Algorithm

Step 1: start up

Step 2: Input: File having no. of cluster and data points.

Output: Coordination of centriod cluster.

Step 3: For j=0 to L(cluster length)
X_axis_centroid=((max_value_X-
min_value_X)/(L+1)*j)+min_value_X
Y_axis_centroid=((max_value_Y-
min_value_Y)/(L+1))*j)+min_value_Y
END For.

Mapper- Function

Input : Data set_file

Output : <K,V> pair // K represent the closet center, // V represent the data point.

Step 1 : if (iteration=first)

Select the centroid through the whole data set.

Otherwise,

Read the reduced based centroid and each

{individual cluster=individual centriod}

Step 2: ED=max_value //ED- represent Eucledian Distance

Step 3: set index=-1

Step 4: For k=0 to L

Do

D= ED(instance, Centers[K]); // D represent distance

If (D<ED)

Do ED = D && index=i

End for

Step 5: Output <K,V> pair

END

When centroid is decided, vectors are clustered and organized into two files, and then the K-Means clustering is accomplished. The map

function is designed in such a way that computes the distance between the vector value and cluster center given into the cluster set. When the mapper function completes its computation, the vector is assigned to the clusters. After the assignment, the centroid of the particular cluster is recalculated [1]. Recalculation of the centroid is done by the reduce function. Once the cluster center is updated and clusters are rewritten with updated values, then one iteration is completed. And then the disk is ready for the next iteration. Algorithms for K-Means implementation on Reducer function are as follows:

Reducer- Function

Input: (K,V) pair // K-represent centroid_cluster // V-represent data points various nodes

Output: (K,V) pair //K-represent index of cluster // V-represent data points value

Step 1: A[i]= datapoints

Step 2: for i=0 to array_last_index(L)

Do

Sum the values

END For.

Step 3: Compute the means of datapoint and add it to file

Step 4: if (new_centroid = previous centroid)

Set converged

Otherwise, go for next iteration

END if

Step 5: Output <K,V> pair

End.

As we discussed in the previous section, HDFS has a strong storage capacity, which can store any kind of data. According to figure 1, here HDFS stores the input file and the output file. Here we use the MapReduce jar file that consists of a class file of k-means, mapper, reducer and one driver file. MapReduce jar file asks ack (acknowledgment) /response to job tracker (Name Node) for the task. At the same time job tracker assign a job ID to that task and send that task with job ID to the task tracker (Data Node), which divides that task into the small subtask. Here, a term used

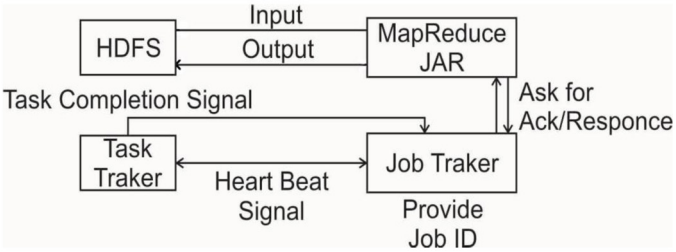

Figure 1

Functioning of MapReduce

a heartbeat signal between job tracker and task tracker. This term has two meaning; one is the positive heartbeat and second is negative heartbeat signal. These are the ack/response signals. Task tracker searches for the resources when got the task with job id from job tracker.

When there is the availability of resource to execute that task then task tracker sends the positive heartbeat signal to the job tracker. And when the resources are not available then it sends the negative heartbeat signal to the task tracker. According to the negative heartbeat signal, it rejects the task due to the unavailability of resources and sends the output as zero to the job tracker. A negative heartbeat signal was used in the traditional MapReduce concept. These days we are using YARN (Yet Another Resource Negotiator), which has a greater memory chunk size of 128 MB as compared to the traditional MapReduce which has 64 MB. In the case of YARN, it uses the scheduling concept. Let's, suppose only 40% of resources are available then it will execute only 40% task, rest 60% task will be waiting for 60% of the resources to be released.

3.2 *Architecture of K-Means on Hadoop framework*

As we discussed in the previous section, we used the MapReduce jar file to run the application. The architecture of the K-means clustering algorithm on MapReduce is shown in figure 2.

We run the K-Means jar on the MapReduce framework that searches the base/driver class. Driver class file decides the centroid and iteration, then it fetches data from HDFS as input and sends that input to the MapReduce programming model. The output of the MapReduce model is then stored in the HDFS.

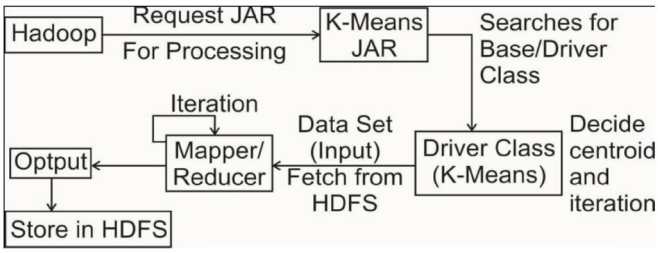

Figure 2

Architecture of K-Means

4. Experimental Result

4.1 *Performance evaluation of K-mean on Hadoop using different individual tools*

For the performance evaluation of K-means clustering algorithms, Ubuntu operating system is used to run the Hadoop framework individually for different tools (Mahout, Spark, R-Hadoop). Table 1 shows the total time taken by K-means algorithm in different ways like K-mean simple (using java codes on MapReduce) that chooses centroids in random fashion, K-means (using java codes on MapReduce) IEC (Initial Equidistant Centres) that chooses centroids on the basis of Euclidian Distance, K-mean on Mahout (using Machine learning library) and K-mean on Spark (Machine learning library). All 4 ways of K-means clustering is individually configured on Hadoop infrastructure either by MapReduce java codes or by Mahout/Spark with the integration of Hadoop. All the experiment is done on one dataset that consist of internet surfing records of static IP address of 4 state of size 2GB.

Table 1

K-mean on Hadoop using different individual tools

K-mean on Hadoop using different individual Tools	Total Time
K-mean (Simple) using Java codes	0:35:45
K-mean (IEC) using Java codes	0:34:01
K-mean on Mahout	0:25:00
K-mean on Spark	0:42:11

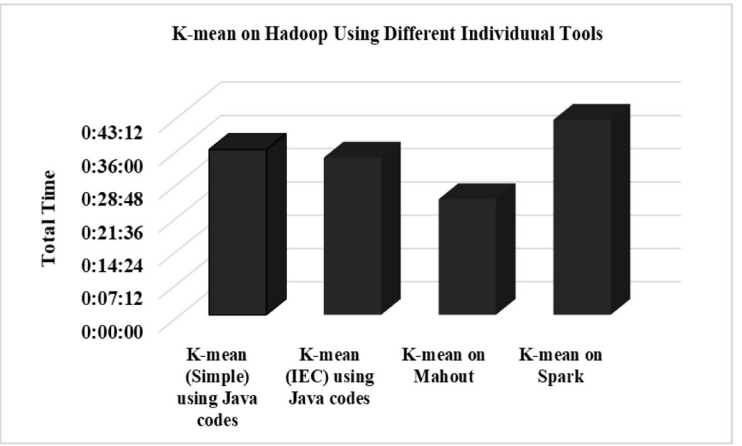

Figure 3

K-mean on Hadoop using different individual tools

The figure 3 shows the individual performance of the K-means clustering over the Hadoop infrastructure. By this experiment it has been noted that, Mahout works faster than the all other ways. Spark is taken longest time to process data. it is because of in-memory process, Spark uses RAM memory to perform any task and due to this Spark is busy to allocate recourses for JobTraker. In same way it is also found that K-mean (Initial Equidistant Centres) is faster than the K-mean simple that chooses the centroids in random fashion.

4.2 *Comparison between K-mean simple and K-mean (with Initial Equidistant Centers)*

In this experiment, both k-means (with Initial Equidistant Centers) and simple K-means run on different iteration levels to form clusters on same static IP address dataset. By the 5-iteration of k-means (with Initial Equidistant Centers), it fetches data from 4 states and each state has one cluster dedicated one static IP address. Records are as follows,

Alaska: IP (192.168.56.91) is used by 2548195 peoples

California: IP (192.168.56.96) is used by 2580868 peoples

Florida: IP (192.168.56.79) is used by 2425142 peoples

Georgia: IP (192.168.56.87) is used by 2904206 peoples

In the same manner, simple K-means computation on 5 iterations fetches only 3 states, each state has one IP address. Records are as follows,

Alaska: IP (192.168.56.91) is used by 2548195 peoples.

California: IP (192.168.56.96) is used by 2580868 peoples.

Florida: IP (192.168.56.79) is used by 2425142 peoples.

By the above results of 5 iterations for both K-means, it is clear that k-means (with Initial Equidistant Centers) fetches accurate results compared to the simple K-Means.

Now, let's increase the iteration level up to 10 iteration of k-means (with Initial Equidistant Centers) and simple K-means. By the 10-iteration k-means (with Initial Equidistant Centers), it fetches data from 4 states and each state has a minimum of 4 clusters, each cluster dedicated one static IP address. Records are as follows,

Alaska:

192.168.56.73 - is used by 3018281 peoples

192.168.56.75 - is used by 3232419 peoples

192.168.56.91 - is used by 2548195 peoples

192.168.56.93 - is used by 3014162 peoples

California:

192.168.56.79 - is used by 3093136 peoples

192.168.56.87 - is used by 3002609 peoples

192.168.56.93 - is used by 3216103 peoples

192.168.56.96 - is used by 2580868 peoples

Florida:

192.168.56.09 - is used by 3154435 peoples

192.168.56.76 - is used by 3212621 peoples

192.168.56.79 - is used by 2425142 peoples

192.168.56.85 - is used by 3188139 peoples

Georgia:

192.168.56.71 - is used by 3094590 peoples

192.168.56.77 - is used by 3008523 peoples
192.168.56.78 - is used by 3192360 peoples
192.168.56.83 - is used by 3018689 peoples
192.168.56.87 - is used by 2904206 peoples
192.168.56.89 - is used by 3118420 peoples

In same manner simple K-means computation on 10 iterations fetches 4 states each state has more than one IP address. Records are follows,

Alaska:
192.168.56.73 - is used by 3018281 peoples
192.168.56.91 - is used by 2548195 peoples
192.168.56.93 - is used by 3014162 peoples

California:
192.168.56.79 - is used by 3093136 peoples
192.168.56.87 - is used by 3002609 peoples
192.168.56.96 - is used by 2580868 peoples

Florida:
192.168.56.79 - is used by 2425142 peoples

Georgia:
192.168.56.71 - is used by 3094590 peoples
192.168.56.77 - is used by 3008523 peoples
192.168.56.83 - is used by 3018689 peoples
192.168.56.87 - is used by 2904206 peoples

By the above results of 10 iterations for both K-means, it is clear that k-means (with Initial Equidistant Centres) fetches accurate results compared to the simple K-Means. K-means (with Initial Equidistant Centres) is more capable to create the cluster on the fewer iterations.

5. Conclusion and Future Work

In the present paper, we have discussed the performance of the K-Means on the MapReduce framework using the Hadoop platform. We

validated our results, by running the K-Means clustering algorithm on 4 different ways as well as on different iteration levels. By the experimental result, we can say that K-Means (with Initial Equidistant Centers) clustering algorithm gives better performance when we increase the iteration with compare to simple K-means. On the other hand, K-Means clustering algorithm over Mahout (on Hadoop infrastructure) environment provides a fast and efficient grouping capacity for similar characteristic data as compare Spark (on Hadoop infrastructure). For future enhancement, there is always a scope for better results. K-Means algorithm can give more refined results by the selection of the initial set of centroids in less iteration [16-20].

References

[1] Anchalia, P.P., Koundinya, A.K. and Srinath, N.K., 2013, June. Mapreduce design of k-means clustering algorithm. In *2013 International Conference on Information Science and Applications (ICISA)* (pp. 1-5). IEEE.

[2] Bayardo, R.J., Ma, Y. and Srikant, R., 2007, May. Scaling up all pair's similarity search. In *Proceedings of the 16th international conference on World Wide Web* (pp. 131-140). ACM.

[3] Cui, X., Zhu, P., Yang, X., Li, K. and Ji, C., 2014. Optimized big data K-means clustering using MapReduce. *The Journal of Supercomputing*, *70*(3), pp.1249-1259.

[4] Dean, J. and Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), pp.107-113.

[5] Ekanayake, J., Pallickara, S. and Fox, G., 2008, December. Mapreduce for data intensive scientific analyses. In *2008 IEEE Fourth International Conference on eScience* (pp. 277-284). IEEE.

[6] Fischer, M.J., 2008, August. Evolution of distributed computing theory: from concurrency to networks and beyond. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing* (pp. 460-460). ACM.

[7] Gong, J., 2018. Association feature mining algorithm of web accessing data in big data environment. *Journal of Discrete Mathematical Sciences and Cryptography*, *21*(2), pp.333-337.

[8] Jiang, D., Tung, A.K. and Chen, G., 2010. Map-join-reduce: Toward scalable and efficient data analysis on large clusters. *IEEE transactions on knowledge and data engineering*, *23*(9), pp.1299-1311.

[9] Kerdprasop, K. and Kerdprasop, N., 2010, October. Parallelization of k-means clustering on multi-core processors. In *Proceedings of the 10th WSEAS international conference on Applied computer science* (Vol. 10, pp. 472-477). World Scientific and Engineering Academy and Society (WSEAS).

[10] Kumar, A., Kiran, M. and Prathap, B.R., 2013, July. Verification and validation of mapreduce program model for parallel k-means algorithm on hadoop cluster. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)* (pp. 1-8). IEEE.

[11] Liao, K., Liu, G., Xiao, L. and Liu, C., 2013. A sample-based hierarchical adaptive K-means clustering method for large-scale video retrieval. *Knowledge-Based Systems*, *49*, pp.123-133.

[12] Li, H.G., Wu, G.Q., Hu, X.G., Zhang, J., Li, L. and Wu, X., 2011, January. K-means clustering with bagging and mapreduce. In *2011 44th Hawaii International Conference on System Sciences* (pp. 1-8). IEEE.

[13] Peter, S.J., 2011. Minimum spanning tree-based clustering for outlier detection. *Journal of Discrete Mathematical Sciences and Cryptography*, *14*(2), pp.149-166.

[14] Wang, C.Y., Zhao, W., Liu, Q. and Chen, H.W., 2017. Optimization of the tool selection based on big data. *Journal of Discrete Mathematical Sciences and Cryptography*, *20*(1), pp.341-360.

[15] Zhao, W., Ma, H. and He, Q., 2009, December. Parallel k-means clustering based on mapreduce. In *IEEE International Conference on Cloud Computing* (pp. 674-679). Springer, Berlin, Heidelberg.

[16] Zhao, Y., 2018. Manufacturing personalization models based on industrial big data. *Journal of Discrete Mathematical Sciences and Cryptography*, *21*(6), pp.1287-1292.

[17] Khan, Tayyab, Karan Singh, Mohamed Abdel-Basset, Hoang Viet Long, Satya P. Singh, and Manisha Manjul. "A Novel and Comprehensive Trust Estimation Clustering-Based Approach for Large Scale Wireless Sensor Networks." IEEE Access, vol. 7, pp. 58221-58240, (2019).

[18] Karthikeyan, T., S. John Peter, and S. Chidambaranathan. "Meta clusters through minimum spanning tree-based clustering for performance analysis of students." J*ournal of Discrete Mathematical Sciences and Cryptography* 14, no. 4 (2011): 349-367.

[19] Peter, S. John. "Local Density-based Hierarchical Clustering using Minimum Spanning Tree." *Journal of Discrete Mathematical Sciences and Cryptography* 16, no. 2-3 (2013): 125-137.

[20] Peter, S. John. "Minimum spanning tree-based clustering for outlier detection." *Journal of Discrete Mathematical Sciences and Cryptography* 14, no. 2 (2011): 149-166.