Comprehend the Performance of MapReduce Programming model for K-Means algorithm on Hadoop Cluster

Chitresh Verma, Rajiv Pandey, Devesh Katiyar

Abstract: MapReduce is a programming model used for processing Big Data. There are had been considerable research in improvement of performance of MapReduce model. This paper examines performance of MapReduce model based on K Means algorithm inside the Hadoop cluster. Different input size had been taken on various configurations to discover the impact of CPU cores and primary memory size. Results of this evaluation had been shown that the number of cores had maximum impact of the performance of MapReduce model.

Index Terms: MapReduce, Hadoop, K-means algorithm, Data mining, cluster algorithm, Big Data, Performance evaluation.

I. INTRODUCTION

Big Data is a complex set of data which cannot be processed in traditional system using data analysis software. It requires special type of data storage and processing systems. Thus, special type framework had been developed for Big Data storage and processing. The processing is done by programming model called MapReduce. Some of MapReduce programming model implementation include "Google MapReduce", "Hadoop MapReduce" and "Amazon Elastic MapReduce".

Hadoop is a popular framework for data storage and processing for Big Data. Hadoop is made up of multiple components and these components are Hadoop MapReduce programming model, Hadoop distributed file system and YARN and associated JAR files. Hadoop MapReduce is a programming model that works on two steps data processing. These two steps are Map Stage and Reduce Stage. The Map stage involves the key value pair formation. The Reduce stage involves process the key value as per logic which is inside reducer. The data is stored in special storage systems which is Hadoop Distributed file systems (HDFS).

The K means is data clustering algorithm which helps in recognition of data pattern like shopping pattern of customers. The clustering of similar data pattern helps in identification of hidden natures of data. It works on identification of group on k variables. The result is k clusters with centroids which used for labeling data set.

This paper had been divided into seven sections. The first section discuss introduction of basic technology and algorithm. Second part provides the related work done in MapReduce and Big Data processing. Third section describes Hadoop MapReduce and HDFS in detail. The fourth section describes K Means algorithm and its importance in evaluating the performance. The fifth section discusses the test scenario which had been conduction under K Means algorithm on Hadoop cluster. Sixth section provides the insight of test result. Finally, the seventh section provides the conclusion of the paper and its result after the evaluating of MapReduce programming model and provides future work.

II. RELATED WORK

MapReduce had been used for data analysis in cheap and reliable manner. The MapReduce is extensively used for data processing of Big Data for various fields like ecommerce, e Government, e-Learning and other fields. Similarly, Big Data had been highly investigated topic for investigator from its storage to processing.

MapReduce programming model was original developed by Google Inc. Later on, Apache software foundation developed Hadoop MapReduce which was inspired by Google MapReduce. Similarly, Amazon Company developed its own MapReduce called Elastic MapReduce (EMR) under its Amazon web services. MapReduce programming model try to provide optimal and fast result with parallel allocation of data processing. It works on basic of concept of map stage and reducer stage so that data with different size, type, values and speed of data transmissions are processed concurrently. MapReduce balance the computation and data transmission over the network. Cost of data transmission is more than the computation when data size and complexity are intense. MapReduce concept could be implicated as combination of mapping, shuffling and sorting and reducing of data set. Apache software foundation had made MapReduce implementation under Hadoop framework. Hadoop MapReduce runs applications with big size of data on parallel configured servers on commodity hardware. MapReduce splits the dataset to form key value pairs. The key value pairs are independent chunks of the data which form by map task of MapReduce model.

Revised Manuscript Received on July 05, 2019.

Chitresh Verma is a PhD scholar at Amity Institute of Information Technology, Amity University, Uttar Pradesh, India.

Dr. Rajiv Pandey Senior Member IEEE is a Faculty at Amity Institute of Information Technology, Amity University, Uttar Pradesh, India.

Dr. Devesh Katiyar is Assistant Professor in Department of Computer Science at Dr. Shakuntala Misra National Rehabilitation University, Lucknow, India.

These key value pairs are taken as input in reduce task. The output of reduce task are then stored on the Hadoop distributed file system (HDFS). In case of any failure, the process had been designed to rerun task. MapReduce and HDFS are mostly kept on same nodes to reduce network workload, thus combination needs of data transmission is negligible. There are also two tracker systems called "JobTracker" and "TaskTracker" in each cluster. They deal with scheduling of jobs on DataNode and ensuring the completion of job without failures.

III. HADOOP MAPREDUCE AND DISTRIBUTED FILE SYSTEM

Hadoop is major framework implementing the MapReduce programming model and distributed file system. The Hadoop is build under Apache Foundation. It is open source framework which runs on commodity hardware.

The Hadoop MapReduce is made up of two stages and these stages are map stage and reduce stage. The data is transformed into key value pairs in the map stage. The key values pairs are simple dataset consist of key which is unique to identify the value. The data is fetched from Hadoop distributed file system at various DataNodes.

In reduce stage, the data is processed to merge and sort the data to get desire output. The output is written to the DataNodes for later on retrieval by the user.

The Hadoop distributed file system (HDFS) is a file system which runs at different systems. HDFS is made up of NameNode and DataNodes. The NameNode is master node which co ordinates the data exchange and stores all data location. There is a secondary NameNode which functions if the NameNode fails. The DataNode stores the data in blocks.

IV. K MEANS ALGORITHM

K means algorithm is data clustering algorithm which used to find group of data with similar data pattern. K means algorithm is an unsupervised learning algorithm which divides the dataset into k cluster and its k centers. It is well-established algorithm used for unsupervised learning technique especially in signal processing.

K means algorithm take large set of elements and transform them into k clusters with k number of means. Initial some random means are assigned and they are iterated for convergence correction. Final means are obtained after the no changes occur in their values after the calculation of new mean. The algorithm in simple steps had been provided in below steps.

Step 1: Input D as set of elements
Step 2: Input K as number of desired clusters
Step 3: Assign random values for means (m1, m2…mk)
Step 4: While convergence criteria are static
Step 5: Calculate new mean for each cluster
Step 6: Assign new mean for item ti
Step 7: End while
Step 8: Print K as set of clusters

In listed steps, the k means algorithm takes D set of elements and K number of desired clusters. Randomly assign values for means such that m1, m2 and mk. Calculation of new mean is done until its convergence criteria become static. Final mean K becomes output as final step.

V. EXAMINING K MEANS ALGORITHM IN VARIOUS TEST SCENARIO

The k means algorithm had been implemented and executed on various test scenarios under Hadoop cluster. A Node named as "quickstart.cloudera" on IP address 10.0.2.15 under port number 50010 had been configured with storage capacity of 554.51 GB. 83176.65 MB that is nearly 1.5 % of total storage had been filled with data for inputs. The processor configuration had been set at various counts for the experiment which range from 4, 6, 8 and 16 cores.

Node	Last contact	Capacity	Blocks
quickstart.cloudera (10.0.2.15:50010)	Mon Apr 29 04:34:47 -0700 2019		
554.51 GB	994	83176.65 MB (1.5%)	2.6.0-cdh 5.13.0

Ten random test scenarios had been chosen for evaluation of the MapReduce characteristics. Five test scenarios had been evaluated with varied number of core (processors) variation as shown in figure 2 and other five test scenarios with primary memory / RAM size variation configurations. The instance type "c4.8xlarge" had been chosen which is available from different sets of instance type in Amazon web service as shown in figure 1. It provides a large set of CPU cores variation and it had been for evaluation of MapReduce programming model under k means algorithm. The test scenarios had been executed sample data of various domains.

Instance type	Default vCPUs	Default CPU cores	Default threads per core	Valid number of CPU cores	Valid number of threads per core
c4.large	2	1	2	1	1, 2
c4.xlarge	4	2	2	1, 2	1, 2
c4.2xlarge	8	4	2	1, 2, 3, 4	1, 2
c4.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2
c4.8xlarge	36	18	2	2, 4, 6, 8, 10, 12, 14, 16, 18	1, 2

Figure 1: "c4.8xlarge" Instance type with 4, 6, 8, 16 CPU cores configuration had been used from above available configuration.

Core (Processors) variation had been executed on k means algorithm with various data size. First test scenario involved data size of 10 GB and 4 processors which completed in 53 seconds. Second test scenario involved data size of 10 GB and 6 processors which completed in 47 seconds. Third test scenario involved data size of 10 GB and 4 processors which completed in 53 seconds. Fourth test scenario involved data size of 40 GB and 8 processors which completed in 106 seconds. Fifth test scenario involved data size of 40 GB and 16 processors which completed in 40 seconds.

S.No.	Start Time (HH:MM:SS)	COMPLETION (TIME IN SEC)	DATA SIZE(IN GB)	CORE (COUNT)
1	22:06:57	53	10	4
2	15:08:04	47	10	6
3	06:23:17	43	10	8
4	18:15:34	106	40	8
5	09:19:48	40	40	16

Primary memory / RAM size variation had been executed on k means algorithm with various data size. First test scenario involved data size of 10 GB and 8 RAM which completed in 45 seconds. Second test scenario involved data size of 10 GB and 6 RAM which completed in 46 seconds. Third test scenario involved data size of 10 GB and 5 RAM which completed in 46 seconds. Forth test scenario involved data size of 10 GB and 4 RAM which completed in 47 seconds. Fifth test scenario involved data size of 10 GB and 3 RAM which completed in 48 seconds.

S.No.	Time (HH:MM:SS)	COMPLETION (TIME IN SEC)	DATA SIZE(IN GB)	RAM (IN GB)
1	04:08:55	45	10	8
2	06:08:10	46	10	6
3	08:07:24	46	10	5
4	16:06:38	47	10	4
5	18:05:51	48	10	3

The k means algorithm execution in both types of test case of CPU core and RAM size variation had an impact on completion time.

VI. EXPERIMENTAL ANALYSIS

The previous section examined the different parameters of MapReduce programming model with K means algorithm.

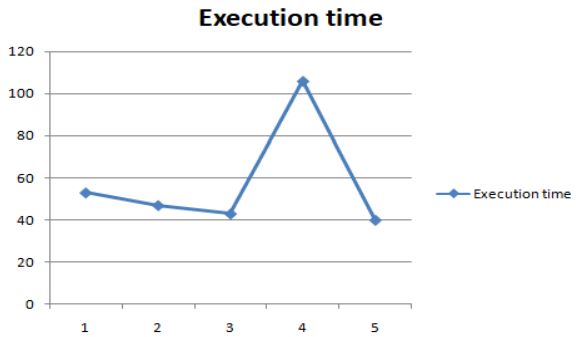

Figure 2: Plot of execution time of five test scenarios of the k means algorithm with various processor configurations.

The execution time of test scenarios under cores variation had been visualized on line chart for understanding the trend in figure 2. Trend shows that processor count is directly proportional to the execution time. Processor count is also directly proportional to data size. The first three case with 10 gigabytes data size shows decrease in execution time with cores increase from 4 to 8. Similarly, fourth and fifth test case with 40 gigabytes data size shows the decrease in execution time. So, the data size is related to the execution time as number of core remains same at 8 cores but the data size is four times with increase in time.

The data size variation with same number of core in the third and fourth test case with 10 gigabytes to 40 gigabytes data size shows increase in execution time that is from 43 seconds to 106 seconds.

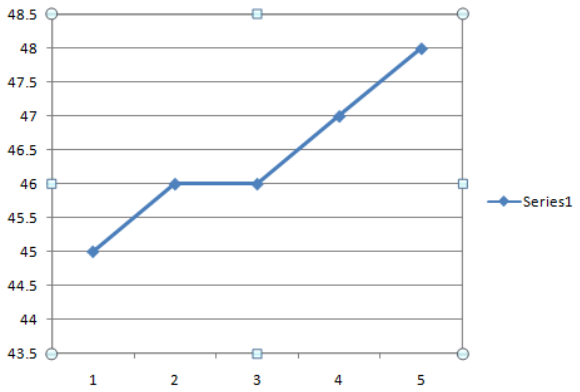

Figure 3: Plot of execution time of five test scenarios of the k means algorithm with various RAM configurations.

The execution time of test scenarios under RAM size variation had been visualized on line chart for understanding the trend in figure 3. Trend shows that RAM size is directly proportional to the execution time. It is also directly proportional to data size.

The impact of processor count is more than the RAM size. On increase and decrease of RAM size had minimum direct impact when compared to processor count.

VII. CONCLUSION AND FUTURE SCOPE

The paper investigated the K means algorithm behavior under MapReduce programming model. Various test scenarios had been tested for k means algorithm in different number of core count and primary memory size configurations. The results after analysis showed the direct relationship between the number of processor and primary memory size under various configurations. On visualization of the data, the trend showed the major impact of number of cores than the memory size. Thus, it could be concluded that processor have most impact on the performance of MapReduce model.

In the future, the power requirement and network load could be assessed in MapReduce based algorithm execution.

REFERENCES

1. "MapReduce Tutorial." The Apache Software Foundation. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html, n.d. Web. 19 June 2019.
2. "Amazon Web Services - Auto Scaling." Tutorials point. https://www.tutorialspoint.com/amazon_web_services/amazon_web_services_auto_scaling.htm, n.d. Web. 14 June 2019.
3. Agrawal, Rakesh, et al. Automatic subspace clustering of high dimensional data for data mining applications. Vol. 27. No. 2. ACM, 1998.
4. Arora, Preeti, and Shipra Varshney. "Analysis of k-means and k-medoids algorithm for big data." Procedia Computer Science78 (2016): 507-512.
5. Davis, Joshua Hoke, et al. "Studying the Impact of Power Capping on MapReduce-based, Data-intensive Mini-applications on Intel KNL and KNM Architectures." arXiv preprint arXiv:1903.11694 (2019).
6. Dhanachandra, Nameirakpam, Khumanthem Manglem, and Yambem Jina Chanu. "Image segmentation using K-means clustering algorithm and subtractive clustering algorithm." Procedia Computer Science 54 (2015): 764-771.
7. Dhanachandra, Nameirakpam, Khumanthem Manglem, and Yambem Jina Chanu. "Image segmentation using K-means clustering algorithm and subtractive clustering algorithm." Procedia Computer Science 54 (2015): 764-771.
8. Eldawy, Ahmed, and Mohamed F. Mokbel. "Spatialhadoop: A mapreduce framework for spatial data." 2015 IEEE 31st international conference on Data Engineering. IEEE, 2015.
9. Gandomi, Amir, and Murtaza Haider. "Beyond the hype: Big data concepts, methods, and analytics." International journal of information management 35.2 (2015): 137-144.
10. Gao, Tilei, et al. "Research on Computing Efficiency of MapReduce in Big Data Environment." ITM Web of Conferences. Vol. 26. EDP Sciences, 2019.
11. Glushkova, Daria, Petar Jovanovic, and Alberto Abelló. "Mapreduce performance model for Hadoop 2. x." Information Systems 79 (2019): 32-43.
12. Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics) 28.1 (1979): 100-108.
13. Hashem, Ibrahim Abaker Targio, et al. "The rise of "big data" on cloud computing: Review and open research issues." Information systems 47 (2015): 98-115.
14. Huang, Zhexue. "A fast clustering algorithm to cluster very large categorical data sets in data mining." DMKD 3.8 (1997): 34-39.
15. Jain, Anil K., and Richard C. Dubes. Algorithms for clustering data. Vol. 6. Englewood Cliffs: Prentice hall, 1988.
16. Krishna, K., and Narasimha M. Murty. "Genetic K-means algorithm." IEEE Transactions on Systems Man And Cybernetics-Part B: Cybernetics 29.3 (1999): 433-439.
17. Kumar, Ravi, et al. "Fast greedy algorithms in mapreduce and streaming." ACM Transactions on Parallel Computing (TOPC)2.3 (2015): 14.
18. Li, Min, et al. "Mronline: Mapreduce online performance tuning." Proceedings of the 23rd international symposium on High-performance parallel and distributed computing. ACM, 2014.
19. Mohd, Wan Maseri Wan, et al. "MaxD K-means: A clustering algorithm for Auto-generation of centroids and distance of data points in clusters." International Symposium on Intelligence Computation and Applications. Springer, Berlin, Heidelberg, 2012.
20. Nayak, Padmalaya, and Anurag Devulapalli. "A fuzzy logic-based clustering algorithm for WSN to extend the network lifetime." IEEE sensors journal 16.1 (2015): 137-144.
21. Qin, Jiahu, et al. "Distributed $ k $-means algorithm and fuzzy $ c $-means algorithm for sensor networks based on multiagent consensus theory." IEEE transactions on cybernetics 47.3 (2016): 772-783.
22. Samiee, Sajjad, et al. "Performance evaluation of a novel vehicle collision avoidance lane change algorithm." Advanced Microsystems for Automotive Applications 2015. Springer, Cham, 2016. 103-116.
23. Santos, Joelson, et al. "Hierarchical Density-Based Clustering using MapReduce." IEEE Transactions on Big Data (2019).
24. Saxena, Ankur, et al. "Handling Big Data Using MapReduce Over Hybrid Cloud." International Conference on Innovative Computing and Communications. Springer, Singapore, 2019.
25. Sengupta, Debapriya, et al. "Modified K-Means Algorithm for Big Data Clustering." 2017 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2017.
26. Srivas, Mandayam C., et al. "Map-reduce ready distributed file system." U.S. Patent No. 9,323,775. 26 Apr. 2016.
27. Stephens, Zachary D., et al. "Big data: astronomical or genomical?." PLoS biology 13.7 (2015): e1002195.
28. Verma, Chitresh, and R. Pandey. "Comparative Analysis of GFS and HDFS: Technology and Architectural Landscape." Proceedings of the2016 IEEE International Conference on Communication Systems and Network Technologies (CSNT '16). IEEE. 2016.
29. Verma, Chitresh, and Rajiv Pandey. "An implementation approach of big data computation by mapping Java classes to MapReduce." Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on. IEEE, 2016.
30. Verma, Chitresh, and Rajiv Pandey. "Analytics-as-a-Service (AaaS)." Exploring Enterprise Service Bus in the Service-Oriented Architecture Paradigm (2017): 26.
31. Verma, Chitresh, and Rajiv Pandey. "Big Data representation for grade analysis through Hadoop framework." Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference. IEEE, 2016.
32. Verma, Chitresh, and Rajiv Pandey and Devesh Katiyar. "Mapping and Classification of Key Features and Technologies behind Distributed File Systems to Make Recognition for Their Inimitable File System." International Journal of Innovative Technology and Exploring Engineering Volume-8 .Issue-8 (2019): 2267–2271. Print.
33. Verma, Chitresh, and Rajiv Pandey. "Mobile Cloud Computing Integrating Cloud, Mobile Computing, and Networking Services Through Virtualization." Design and Use of Virtualization Technology in Cloud Computing. IGI Global, 2018. 140-160.
34. Verma, Chitresh, Rajiv Pandey, and Devesh Katiyar. "Performance Evaluating System Based on MapReduce in Context of Educational Big Data." International Journal of Organizational and Collective Intelligence (IJOCI) 8.1 (2018): 1-12.
35. Yadav, Santosh, and Jay Prakash. "A Survey on Implementation of Word-Count with Map Reduce Programming Oriented Model using Hadoop Framework." Available at SSRN 3351074 (2019).
36. Zaharia, Matei, et al. "Apache spark: a unified engine for big data processing." Communications of the ACM 59.11 (2016): 56-65.

Zheng, Haoyang, Yong Deng, and Yong Hu. "Fuzzy evidential influence diagram and its evaluation algorithm." Knowledge-Based Systems 131 (2017): 28-45.

AUTHORS PROFILE

Chitresh Verma is a PhD scholar at Amity Institute of Information Technology, Amity University, Uttar Pradesh, India. He has multiple publishication including IEEE Conferences, ACM indexed journal and Scopus indexed Journal. He has also author three chapters published by IGI Global, USA.

Dr. Rajiv Pandey Senior Member IEEE is a Faculty at Amity Institute of Information Technology, Amity University, Uttar Pradesh, India. He has multiple publications in reputed journals and conferences. He is also member of various computer societies.

Dr. Devesh Katiyar is Assistant Professor in Department of Computer Science at Dr. Shakuntala Misra National Rehabilitation University, Lucknow, India. His research areas are Software Engineering, Data Mining & Data Warehouse. He has many years of teaching experience.