DOI: http://dx.doi.org/10.26483/ijarcs.v8i9.4797

ISSN No. 0976-5697

Volume 8, No. 9, November-December 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

ENHANCEMENT ON TRAFFIC AWARE PARTITION IN MAPREDUCE USING CLUSTERING TECHNIQUES

T.Sindhupriya
Research Scholar
PG & Research Department of Computer Science
Dr.N.G.P Arts and Science College (Autonomous)
Coimbatore, India

Dr .R.Kousalya
Professor and Head
Department of Computer Applications
Dr.N.G.P Arts and Science College (Autonomous)
Coimbatore, India

Abstract: The Map reduce is a programming model for handling and processing the huge datasets using map and reduce tasks in parallel distributing. To increase the execution of map reduce many number of activities have been made, but they ignore to deal with network traffic produced in shuffle stage. The existing map reduce traffic-aware partitions suffer from partition skew issue, where the output of map tasks is unevenly distributed among reduces tasks. Existing arrangements take after a comparative rule that repartitions workload among diminish undertakings. In any case, those methodologies frequently cause elite overhead because of the segment estimate expectation and repartitioning. The proposed work chooses dynamic data aware parallel with k-Means algorithm (DDAP-kM), a framework that provides dynamic partitioning skew reduction and clustering map reduce jobs. These works cope with partitioning skew by adjusting runtime resource allocation to reduce tasks. By the experimental results network traffic cost is compared in terms of traffic aware partition algorithm and DDAP-kM algorithm.

Keywords: Big data; Map reduce; Dynamic parallel K- means algorithm; Network Traffic Reduction

1. INTRODUCTION

Huge information is a term for informational indexes that are so substantial and deficient to manage conventional information handling. Big data refers to the exponential growth and availability of the data. The data volume, data variety and data velocity are the three dimensions limit [1]. Big data is a huge amount of structured, semi-structured and unstructured data that has been extracted for information. The huge data features are prevalent and unobtrusive dealing with control, data joining and quality capacities, unstructured substance organization. Social database organization structure is assorted to work with gigantic data. Hadoop is used to beat this issue. Hadoop is a programming structure used to help the handling of extensive informational collections in a disseminated figuring condition. Hadoop accomplishes finish parallelism for capacity and conveyed figuring utilizing Map Reduce [2]. Hadoop composed and based on two free system HDFS and Map Reduce. HDFS (Hadoop Distributed File System) is a dependable circulated document framework that gives high throughput access to information [3]. Mapreduce (processing) is a framework for performing high performance distributed data processing using the divide and aggregate programming paradigm. A Map divides the information into individual chunks which are processed by map jobs in parallel. The output of map framework are then input of the reduce tasks. The each input and output of the role is stored in a file system. If users specify a map function that process a key/value pair to get a collection of intermediate key/value pairs and a reduce function that manages all intermediate values related to equivalent intermediate key. The Massive Parallel handling system for preparing information is circulated on a product group. Essentially it is a parallel preparing procedure for handling information as opposed to doing it serially which unquestionably spares time.

2. RELATED WORK

Many researchers have been done regarding the map reduce job and performance improvement of map reduce job. Huan ke have proposed “On Traffic-Aware Partition and Aggregation in Map reduce for big data Applications” [4]. In this paper, to affect massive scale optimization issue for large information applications distributed algorithm is proposed and on-line algorithm additionally designed to regulate the data aggregation and data partition in a dynamic way. Lee, Youngseok propose "An internet traffic analysis method with map reduces” [5]. In this paper an online flow analysis methodology supported the Map Reduce software framework of the cloud computing platform for a large-scale network. Fan, Liya, et al "Improving the load balance of map reduces operations based on the key distribution of pairs” [6] have composed the calculation of key conveyance among the transitional key esteem match to enhance the heap adjusting in the guide decrease. Hsueh, Sue-Chen, Ming-Yen Lin, and Yi-Chun Chiu have proposed "A load-balanced map reduce algorithm for blocking-based entity-resolution with multiple keys” [7] is produced viable load adjusting for outline, however in this characterized the sum total of what above have been done an exploration, and by considering the heap adjusting in the guide decrease, the expansive measure of information is straightforwardly exchange to the reducer to give the last yield to the client. Condie et al “Map reduce Online” [8] acquaints the combiner work with consolidate the middle of the road date with key an incentive from the guide and measure of information to be joined and sent to the reducer.

3. FRAMEWORK

Maps input key/value pairs to a set of intermediate key/value pairs. Maps are individual tasks that transform input records into intermediate records [9]. Mapper fundamentally work in parallel so we call that output as intermediate records, now hadoop manages the information in form of key value so these records are key and value. After mapper finishes there is one more phase in map reduce that is called shuffling and sorting. So shuffling is basically movement of intermediate records from mappers to reducers [10]. So, the process of exchanging the intermediate output from the map tasks to where they are required by the reducers is known as shuffling. All of the values with the same key are presented to a single reducer together, so reducer works on one set of records at time and it gets key and list of values. The local input data have been pre-loaded then mapper generates the intermediate data which we have seen in our index cards then values are exchanged by shuffle process reducing process then generates the output. The output is stored locally. Map reduce is the execution engine of Hadoop. Its duty is to get the jobs executed [11].

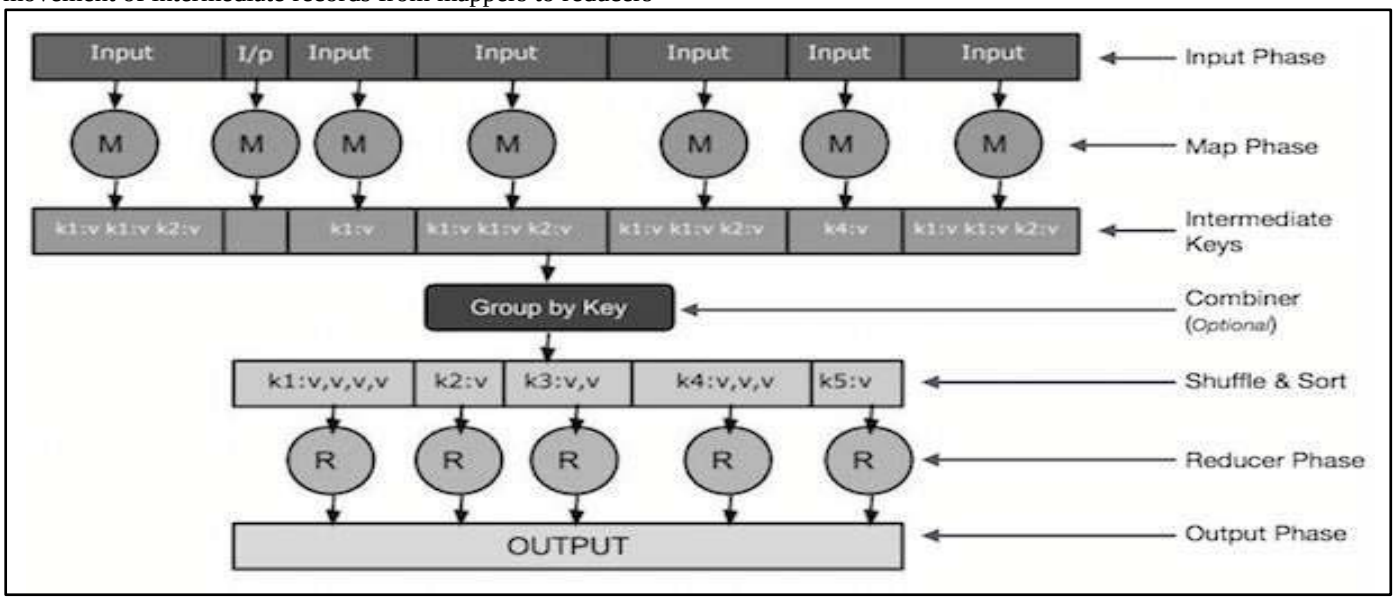

Figure 1: Map reduce Flow

There are two main components of Mapreduce: job tracker and the task tracker. The job tracker is hosted inside the master and it receives the job execution request from the client. Its main duties are to big computations in small parts. Allocate the partial computations that are tasks to slave nodes monitoring the progress and report of task execution from the slave. The main storage component of the Hadoop Framework is HDFS [12]. HDFS is framed for processing and maintaining the huge datasets efficiently among cluster nodes. The computation infrastructure of hadoop has to cooperate with map reduce; the data has to be uploaded to HDFS from local file systems. However, when the data size is large, this upload procedure takes more time, causing delay for key tasks [13].

4. PROPOSED WORK

Mapreduce is programming model for building parallel data processing applications in the cloud demonstrate. The existing MapReduce traffic-aware partition suffers from partition skew issue; where the output of map tasks is unevenly disseminated among reduce tasks [14]. Existing solutions follow a similar principle that repartitions workload among reduce tasks. These approaches often incur high performance overhead due to the partition size prediction and repartitioning. The proposed technique dynamic data aware parallel with k-Means algorithm (DDAP-kM), a framework that gives a dynamic partitioning skew mitigation and clustering map reduce jobs. Rather than repartitioning workload among reduce tasks, the proposed partitioning skew problem by controlling the amount of resources allocated to each reduce task. This approach completely eliminates the repartitioning overhead, yet is simple to implement. There are six fundamental segments: Partition Size Monitor, running in the Node Manager; Partition Size Predictor, Task Duration Estimator, and Resource Allocator, running in the Application Master, Fine-grained Container Scheduler, running in the Resource Manager. Each Partition Size Monitor records the measurements of middle of the road, at last grouping input dataset in light of parallel k implies bunch calculation [15]. A guide undertaking creates the information at run-time and sends them to the Application Master through pulse messages. The Partition Size Predictor gathers the parcel measure reports from Node Managers and predicts the segment sizes of each diminish errand for this activity. The Task Duration Estimator develops measurable estimation model of decrease errand execution as an element of its segment size and asset designation. That is, the length of a lessen undertaking can be evaluated if the parcel size and asset designation of this assignment are given. The Resource Allocator decides the measure of assets to be designated to each diminish errand in light of the execution estimation. In conclusion, the Fine-grained Container Scheduler is in charge of booking assets among all the Application Masters in the bunch, in view of planning strategies, for example, reasonable booking and Dominant Resource Fairness (DRF). Note that the schedulers in unique Hadoop accept that all diminish assignments have homogeneous asset prerequisites as far as CPU and memory [16]. Be that as it

may, this isn't suitable for Map Reduce occupations with apportioning skew. We have altered the first schedulers to help fine-grained holder booking that enables each undertaking to ask for assets of adjustable size.

5. ALGORITHM

A. Dynamic Data Aware Parallel with K-Means Algorithm (DDAP-kM)

i. The first step partition size prediction models which will forecast the partition sizes of reduce tasks at run-time. Specifically, it will accurately predict the scale of every partition once solely 100 percent of map tasks have completed.
ii. The second step establish a performance task model which correlate the finishing time of individual reduce tasks with their partition sizes and resource allocation.
iii. The third step scheduling algorithm that dynamically adjusts resource allocation to each reduce task using our task performance model and the estimation of the partition size. This can reduce the operation time variation among reduce tasks that have dissimilar sizes of partitions to practice, thereby accelerate the job finishing point.
iv. Final step cluster the data based on parallel k-means clustering algorithm.

B. Parallel k-Means Cluster

The algorithm randomly select k object from the whole objects that represents the initial cluster centers. In k-means algorithm, the distance calculation has been calculated [17]. In each iteration, it would need a whole of (nk) distance computation where n is the number of objects and k is the number of clusters being created.

Algorithm: 1
Map (key, value)
Input:
Center variable, offset key, sample value
Output:
<Key, value>pair where key- the index of the closest center point and value-a string include of model information
Step 1: create the value of model illustration;
Step 2: minDis = Double.MAX VALUE;
Step 3: index = -1;
Step 4: For i=0 to centers. Length do
Dis= ComputeDist (instance, centers[i]);
If dis < minDis {minDis = dis; index = i;}
Step 5: End For
Step 6: Take index as key;
Step7: create value as a string comprise of different dimensions;
Step 8: output < key, value>pair;
Step 9: End
Algorithm: 2
Reduce (key, V)
Input:
Key is the index of the cluster; Vis the list of the partial sums from different host
Output:
<Key, value>pair: where key -index of the cluster, value - string representing the new center
Step 1: Initialize one array record the sum of value of each dimensions of the samples contained in the same cluster,
Step 2: Initialize the offset NUM as 0 to the sum of sample number in the similar cluster;
Step 3: while (V.hasNext ())
{Create the model instance from V.next ();
Add the values of different dimensions of instance to the array NUM += num;
Step 4 :}
Step 5: Divide the entries of the array by NUM to get the new center's coordinates;
Step 6: Take key as key';
Step 7: Construct value' as a string comprise of the center's coordinates;
Step 8: Output< key, value>pair;
Step 9: End

6. EXPERIMENTAL RESULTS

The dynamic data aware parallel with K-Means algorithm (DDAP-kM) on Hadoop 1.2.1 is an additional feature. Parallel k-Means (PkM): This application classifies movies based on their ratings using the Netflix movie rating data. The starting values of the cluster centroids provided by PUMA and run on iteration.
Validate the accuracy of the partition size prediction model. Execute the Map Reduce jobs on dissimilar datasets, and work out the mean total percentage error (MAPE) of all partitions in every state. The MAPE is defined as follows

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|P_i^{pred} - P_i^{measrd}\right|}{P_i^{measrd}}$$

Where N is the amount of reduce tasks in a job, P_i^{pred} and P_i^{measrd} are the predicted and calculated value of partition size of reduce task i, respectively [18].

Table 1: Comparisonon Number of job ID and Time taken in seconds

Algorithm	Sequential ID				
	200	**400**	**600**	**800**	**1000**
Traffic-Aware Partition (in seconds)	141	185	244	289	312
DDAP-kM (in seconds)	89	147	181	265	297

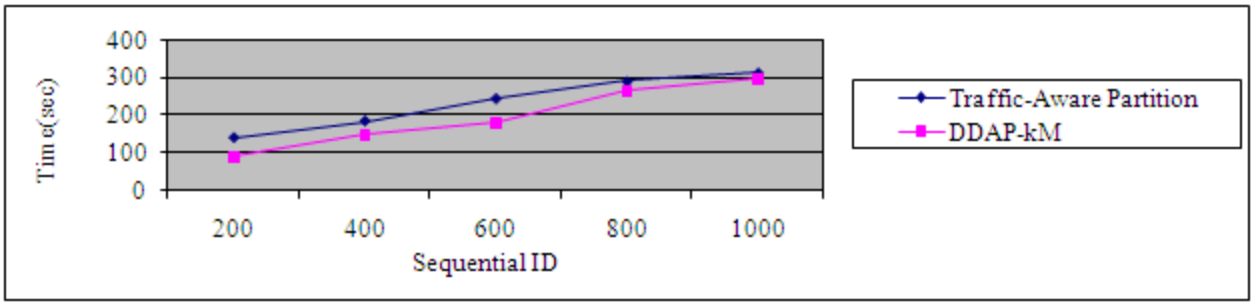

Figure 2: Comparison on Number of job ID and Time taken in seconds

Table 2: Comparison on job completion time between Hadoop platforms

Algorithm	Sequential ID				
	YARN	**LATE**	**Skew Tune**	**Traffic-Aware Partition**	**DDAP-kM**
K Means	685	967	843	532	437
Word count	865	1036	745	671	430

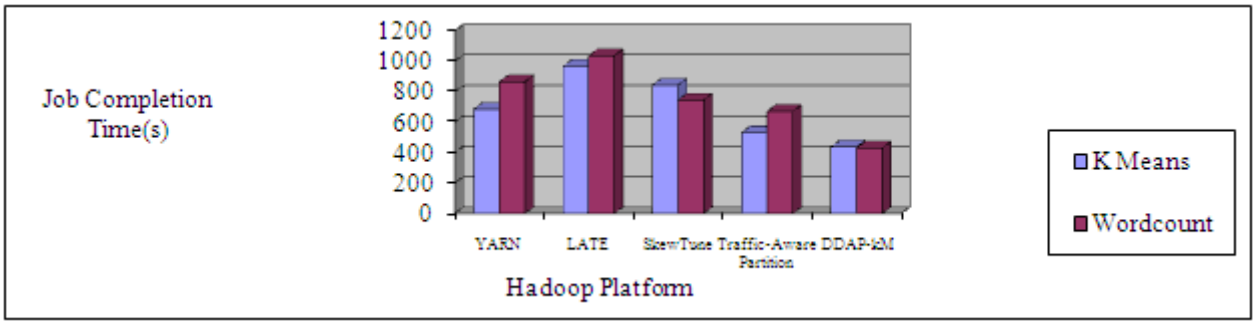

Figure 3: Comparison on job completion time between Hadoop platforms

Table 3: Comparison on Traffic rate and Number of Map tasks

Algorithm	No of Map Task				
	2	4	8	10	12
Traffic-Aware Partition	1.45	2.36	3.97	4.69	6.43
DDAP-kM	0.56	1.45	2.66	3.79	4.32

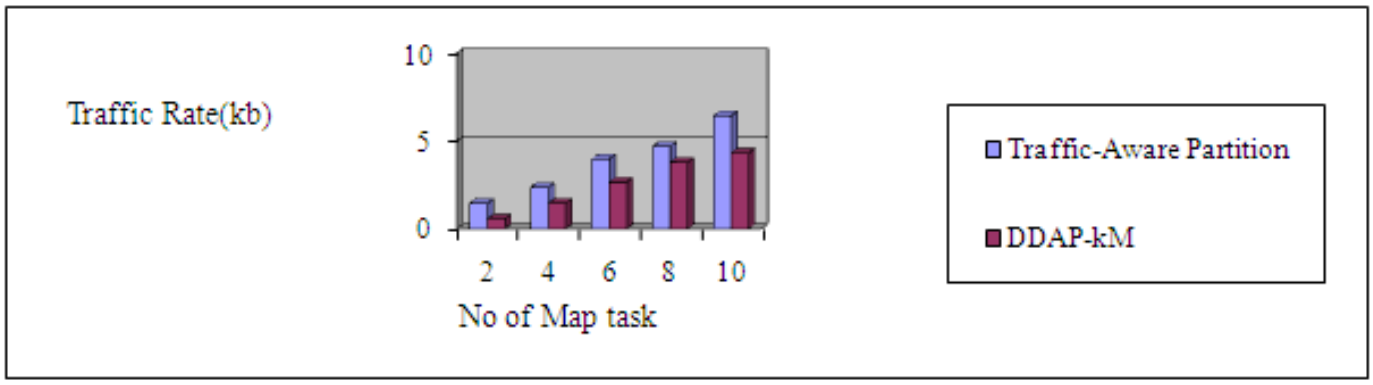

Figure 4:Comparison on Traffic rate and Number of Map tasks

Table 4:Comparison on Traffic rate and Number of Reduce tasks

Algorithm	No of Map Task				
	2	4	8	10	12
Traffic-Aware	2.67	2.12	1.89	1.46	1.34

Partition					
DDAP-kM	2.33	1.78	1.56	1.12	0.97

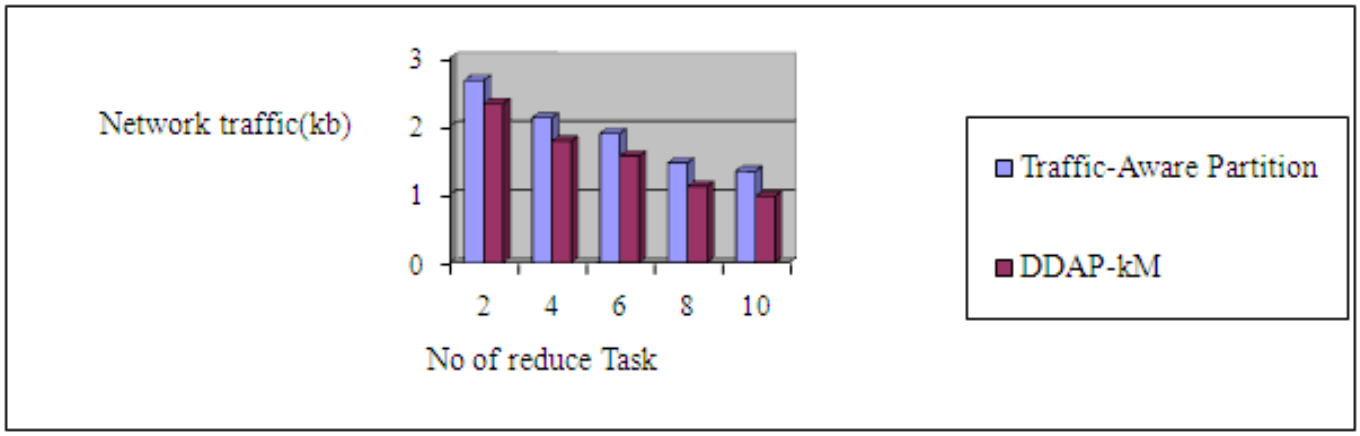

Figure 5: Comparison on Traffic rate and Number of Map tasks

7. CONCLUSION AND FUTURE WORK

Mapreduce is a programming model for processing parallel information in the cloud display. The current Mapreduce movement aware of dividing skew issue; where the yield of guide undertakings is unevenly circulated among diminishing tasks. Existing arrangements take after a comparable rule that repartitions workload among decrease tasks. These methodologies regularly bring about best overhead because of the measure expectation and repartitioning. The proposed technique dynamic information mindful parallel with k-Means calculation (DDAP-kM), a structure that gives dynamic dividing skew relief and grouping map decrease occupations. Rather than repartitioning workload among distributed tasks, the proposed dividing skew issue by controlling the measure of assets dispense to each decrease assignment. This approach totally dispenses with the repartitioning overhead, yet is easy to actualize. The future work limits client endeavors in skew reduction through better programming interfaces and execution models utilizing static and dynamic examination or improvements in multi-inhabitant conditions.

8. REFERENCES

[1] Katal, Avita, Mohammad Wazid, and R. H. Goudar. "Big data: issues, challenges, tools and good practices." Contemporary Computing (IC3), 2013 Sixth International Conference on. IEEE, 2013.

[2] Dittrich, Jens, and Jorge-Arnulfo Quiané-Ruiz. "Efficient big data processing in Hadoop MapReduce." Proceedings of the VLDB Endowment 5.12 (2012): 2014-2015.

[3] Patel, Aditya B., Manashvi Birla, and Ushma Nair. "Addressing big data problem using Hadoop and Map Reduce." Engineering (NUiCONE), 2012 Nirma University International Conference on. IEEE, 2012.

[4] Ke, Huan, et al. "On traffic-aware partition and aggregation in mapreduce for big data applications." IEEE Transactions on Parallel and Distributed Systems 27.3 (2016): 818-828.

[5] Lee, Youngseok, Wonchul Kang, and Hyeongu Son. "An internet traffic analysis method with mapreduce." Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP. IEEE, 2010.

[6] Fan, Liya, et al. "Improving the load balance of mapreduce operations based on the key distribution of pairs." arXiv preprint arXiv:1401.0355 (2014).

[7] Hsueh, Sue-Chen, Ming-Yen Lin, and Yi-Chun Chiu. "A load-balanced mapreduce algorithm for blocking-based entity-resolution with multiple keys." Proceedings of the Twelfth Australasian Symposium on Parallel and Distributed Computing-Volume 152. Australian Computer Society, Inc., 2014.

[8] Condie, Tyson, et al. "MapReduce online." Nsdi. Vol. 10. No. 4. 2010.

[9] Khalil, Salma, et al. "Mapreduce performance in heterogeneous environments: a review." International Journal of Scientific & Engineering Research 4.4 (2013): 410-416.

[10] Steele, Brian, John Chandler, and Swarna Reddy. "Hadoop and MapReduce." Algorithms for Data Science. Springer International Publishing, 2016. 105-129.

[11] Kiran, M., et al. "Verification and Validation of MapReduce Program Model for Parallel Support Vector Machine Algorithm on Hadoop Cluster." International Journal of Computer Science Issues 10.1 (2013): 317-325.

[12] L. Fan, B. Gao, X. Sun, F. Zhang, and Z. Liu, "Improving the load balance of mapreduce operations based on the key distribution of pairs," arXiv preprint arXiv:1401.0355, 2014.

[13] Le, Yanfang. Datacenter-Network-Aware Online Load Balancing in MapReduce. Diss. Applied Sciences: 2015.

[14] Wang, Weina, et al. "Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality." IEEE/ACM Transactions on Networking 24.1 (2016): 190-203.

[15] Yang, Yang, Xiang Long, and Bo Jiang. "K-means method for grouping in hybrid mapreduce cluster." Journal of Computers 8.10 (2013): 2648-2655.

[16] Yuanquan, Fan, et al. "Improving MapReduce performance by balancing skewed loads." China Communications 11.8 (2014): 85-108.

[17] Gu, Rong, et al. "SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters." Journal of parallel and distributed computing 74.3 (2014): 2166-2179.

[18] Gao, Yufei, et al. "Handling Data Skew in MapReduce Cluster by Using Partition Tuning." Journal of Healthcare Engineering 2017 (2017).