International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-06, Issue-10, JAN 2021

Ensemble Learning assisted Hadoop big data optimization for application speed up

Nandita Yambem, Research Scholar,Dept of CSE ,VTU-RRC Bangalore, India,
nanditayambem@gmail.com
Dr A.N.Nandakumar, Professor ,Dept of CSE , City Engineering College ,Bangalore, India,
inandakumar53@gmail.com

***Abstract*— Hadoop has become a default platform for big data applications. Hadoop has many tunable parameters which have very big influence on the performance of the map reduce applications. Finding the best value for these tunable parameters has been an important research area. Many heuristic solutions have been proposed for tuning these performance parameters. Most of the heuristics solutions are based on the exploiting linear relationship based on intermediate data generated alone without consideration for the environment and application nature. Thus a adaptive learning based solution is needed to fine tune the Hadoop parameters. In this work, we apply adaptive ensemble of machine learning techniques to fine tune the Hadoop configuration parameters based on the past history of execution of the jobs.**

Keywords: Hadoop , SVM , KNN ,Neural Network , Regression .

I. INTRODUCTION

Hadoop is becoming a popular platform for big data applications. Hadoop is based on map reduce architecture. The data to be processed in kept in Hadoop distributed file system. Applications are written in form of Map and Reduce. Map processes the input data and generates intermediate data of key and value pairs. The reduce job process aggregates the intermediate data based on keys and generates the results in form of key value pairs. Hadoop framework executes multiple map reduce instances in parallel by mapping them to threads in the OS. Hadoop is becoming a popular platform for big data applications. Hadoop is based on map reduce architecture. The data to be processed in kept in Hadoop distributed file system. Applications are written in form of Map and Reduce.

Intermediate data volume has been used as a important criteria for deciding the number of maps and number of reduce in many previous works as discussed in literature review. All these works are based on the observation that without sufficient number of reduces tasks, the volume of intermediate data grows high and it may result in system crash. So by increasing the number of reduce tasks in proportion to intermediate volume, the system crash due to insufficient memory could be avoided. The relation between the intermediate data volume and the number of reduce tasks is modeled as linear relation in many previous works. But considering the nature of applications, this linear relation does not hold good. Especially in a heterogeneous environment, linear relationship between intermediate data and number of reduce tasks is no longer valid. To mitigate intermediate memory build surge, the number of maps are controlled in these approaches which results in slow down of applications. In this work, we explore the use of machine learning models to optimize the Hadoop configuration parameters and solve the issues in linear modeling based on intermediate data volume. Adaptive Ensemble of machine learning models is used for fine tuning the Hadoop configuration parameters. Hadoop jobs are profiled to collect the execution profile of map and reduce tasks, average rate of intermediate data processed etc and based on these parameters, machine learning models are built to predict the number of maps and reduce. The predicted values are configured in Hadoop run time system to increase the application performance.

II. SURVEY

Authors in [1] proposed solution for increasing the efficiency of Hadoop Distributed File System (HDFS) to improve the application speedup in Hadoop. By increasing the efficiency of HDFS, the data access concurrency between the parallel execution of map and reduce tasks is improved and this results in application speedup. The validity of this solution was tested in Grid'5000 dataset. But the gain received is comparatively lower (less than 5%) than the optimizing Hadoop configuration parameters. Authors in [2] used data compression as a strategy for improving Hadoop performance. The data movement between HDFS and the applications is done in a compressed way, so that the net bandwidth between the data intensive applications and HDFS is reduced. The validity of this approach was tested

 DOI : 10.35291/2454-9150.2021.0017

against HTRC word counting task in XSEDE resources. Data compression strategy is not suitable for all kinds of applications and performance gain is less than 5%. Authors in [3] proposed a scheduling algorithm called, Longest Approximate Time to End (LATE) to increase the performance of Hadoop. Finish time of the jobs are estimated and job which hurts the response time the most is executed first. By this way lingering jobs are cleared and their impact on application slowdown due to memory occupancy is cleared. Estimating the finish time accurately is a challenge in heterogeneous environment. A phase level scheduling algorithm is proposed in [4] to increase the application speedup. The jobs are divided to phases at a fine-grained level and phase level scheduling is done. By this way parallelism is increased. The validity of this solution was tested against a 10 node Hadoop cluster and phase level scheduler was able to increase the performance by 1.3 times. A major problem in this approach is that not all applications are suitable for phase level scheduling and gain is lower considering the resource concurrency. Authors in [5] applied data locality to increase the application speedup. Data locality reduces the cross-switch network traffic. By this performance degradation due to communication can be reduced. Various data locality models were explored in this work for single cluster and cross cluster environment. For efficient data locality the communication between the tasks must be known in advance, which is not possible for all kinds of applications. Authors in [6] used intermediate data compression as a strategy to increase the application speedup. Due to compression of intermediate data, the shuffling overhead is reduced and computational cycles saved in shuffling overhead can be used for applications. Authors also proposed a lossless compression scheme for intermediate data which is able to reduce the intermediate data volume by five times. But the overhead in decompressing and using the data in reduce operations make this approach not suitable for all the applications. Authors in [7] proposed a strategy to decide when to compress the intermediate data. A decision algorithm is developed which helps the users to decide when to use compression. A significant achievement in this solution is that it is able to achieve 60% energy saving. Authors in [8] proposed a efficient data shuffling algorithm to increase the application speedup in Hadoop. Data movement is made effective using the proposed shuffling algorithm, thereby reducing the power consumption and extra computing cycles needed for shuffling. These extra computing cycles can be used for applications for increasing the speedup. But the performance gain is less than 10%. Authors in [9] used data compression to reduce the bandwidth need for applications in fetching the data. The validity of the solution was tested against HTRC word counting task and large scale medical image dataset. The application speed up is less than 5%, but there is a significant reduction in network bandwidth and energy consumption due to data compression. Authors in [10] modified the vanilla version of Hadoop with a fast intermediate layer to increase the application speedup. This intermediate layer is fault tolerant and optimized for concurrency. By reducing the read/write time for intermediate data, the overall execution time of jobs in increased. But the gain is lower than 10%. Authors in [11] optimized memory allocation, overhead in garbage collection, in-memory increase to increase the application speedup. Memory optimizations are able to increase the speed up by 2 times. Intermediate data access is reduced by moving to in-memory instead of disk, thereby increasing the application speedup but the resource cost is very high in this approach. Authors in [12] classified the Hadoop optimization parameters. The parameters are classified to different categories of IO-tuning, CPU-tuning, both-tuning and none. Depending on the application nature of CPU intensive or IO intensive, the corresponding set of parameters to be optimized is selected. For CPU intensive applications, CPU-tuning parameters are selected. For IO intensive applications, IO-tuning parameters are selected. Though the work identified the parameters to be tuned based on application nature, it did not propose any method to optimize the corresponding parameters. Authors in [13] proposed application collocation as a strategy to increase the application speedup. Through way of collocation, the unnecessary network overhead and the energy consumption due to it can be reduced. Co-location is able to increase the speed-up by 8%. Authors in [14] increased the speed up for failed tasks in Hadoop by task failure resilience. The failed tasks are executed from last known execution point instead of execution from start. Though it results in overhead for check pointing, the speed up is increased. Authors in [15] used genetic algorithm based search to identify the parameter settings to achieve near optimal execution time. The approach is complex when applied to real world environment as the multiple parameters interaction is not considered in this work. Authors in [16] proposed a memory model for application speedup. Based on the history of job executions, the total memory required for the task is pre-allocated. But the solution has reduced throughput. Authors in [17] proposed a noisy gradient approach for fine tuning the Hadoop configuration parameters. The approach is not adaptive.

III. RESEARCH GAP

The current solution for increasing the application speed up in Hadoop platform can be categorized as

1. Data compression
2. Memory allocation
3. Hadoop configuration parameter tuning

The gain is low in case of data compression and memory allocation based approaches. Hadoop configuration parameter tuning approaches provides higher performance gains. But following are some of the problems in the existing Hadoop configuration tuning approaches

 DOI : 10.35291/2454-9150.2021.0017

1. There are not adaptive to application nature
2. They consider many parameter to optimize and they don't consider the interaction due to it
3. They are mostly based on linear relationship of intermediate data and execution time which does not apply in heterogeneous environment.

This work addresses the above three challenges.

IV. PROPOSED SOLUTION

The proposed solution uses adaptive ensemble of machine learning to solve the three problems identified in the existing Hadoop configuration tuning approaches. The proposed model uses three different machine learning models of

1. SVM Regression
2. KNN
3. Neural Network

These three models predict the optimal number of reduce and map needed for execution of the job. The results of each of above classifiers are ensembled in a adaptive weighted manner. The weights for ensembling is learnt dynamically to suit for different application needs.

A. *Profiling*

Jobs are executed in Hadoop by varying the number of maps and number of reduce and following parameters are collected.

1. Job completion time (P1)
2. Intermediate data size (P2)
3. Average map execution time in a slot(P3)
4. Average reduce execution time in a slot(P4)
5. Average IO wait time(P5)
6. Number of maps (P6)
7. Number of reduces (P7)
8. Average Intermediate data build up rate(P8)

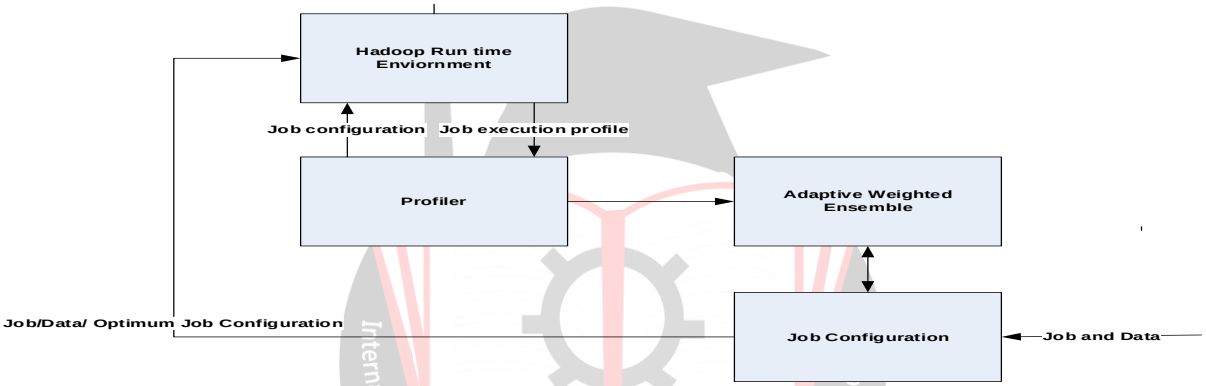

Figure 1: Proposed Architecture

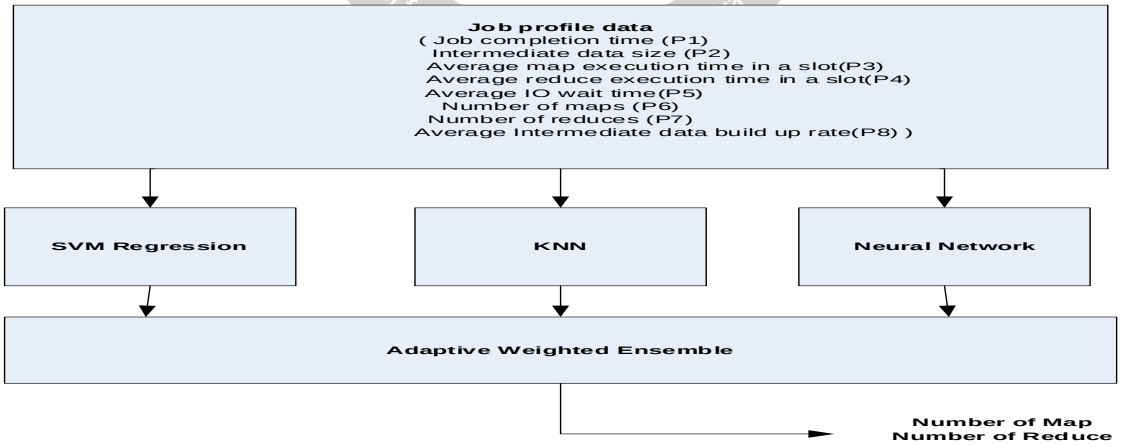

Figure 2 : Adaptive weighted ensemble model

B. *SVM Regression*

Two SVM regression model is constructed with one model for predicting number of maps and another for predicting number of reduces. The SVM regression model for prediction of number of maps takes – job completion time, intermediate data, average map execution time in a slot, average reduce execution time in a slot , average IO wait time as input and number of maps as output. The SVM regression model for prediction of number of reduce takes – job completion time, intermediate data, average map execution time in a slot, average reduce execution time in a slot ,

 DOI : 10.35291/2454-9150.2021.0017

average IO wait time as input and number of reduce as output.

C. *KNN*

Taking 80% of the profiled data as training data, KNN model is built for predicting the number of maps and reduce. The optimum value of K is found using elbow method. In elbow method a cost function is defined and the value of it for various values of K is found and plotted. The value of K at which a elbow is created in the cost is decided as the optimal value of cost. The cost function is calculated as average of distance of each point in cluster to its cluster centroid. It is calculated as

$$cost = \frac{\sum_{i=1}^{k}\sum_{j=1}^{|cluster\ i|}|j - centroid_j|}{K}$$

Where K is the total number of clusters.

The cost is plotted for various value of K

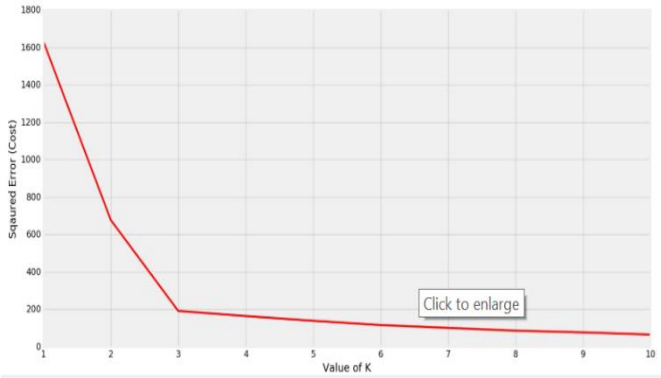

Figure 3: Elbow results

Elbow is created at k=3, so the optimal value for k is decided as 3.

D. *Neural Network*

Two neural network (NN) model is constructed for prediction of number of maps and reduces. The neural network for prediction of number of maps is constructed with following parameters

Parameters	Values
Inputs	P1,P2,P3,P4,P5,P8
Ouput	P6
Number of layers in neural network	3
Number of neuron is layer 1	6
Number of neurons in layer 2	15
Number of neuron in layer 3	1
Activation function	Relu

Table 1 : Neural Network for Map

The neural network for prediction of number of reduce is constructed with following parameters

Parameters	Values
Inputs	P1,P2,P3,P4,P5,P8
Output	P7
Number of layers in neural network	3
Number of neuron is layer 1	6
Number of neurons in layer 2	15
Number of neuron in layer 3	1
Activation function	Relu

Table 2: Neural Network for Reduce

The neural network is learnt with back propagation learning algorithm. The accuracy of the neural network for the training and validation is as given below.

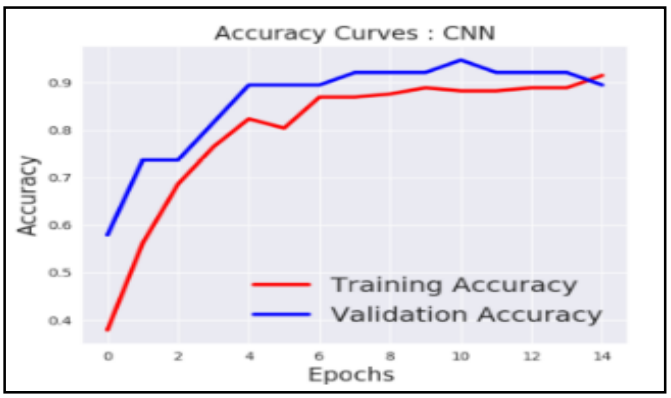

Figure 4: Neural Network accuracy

E. *Adaptive weighted Ensemble*

An Ensemble of the results of SVM regression, KNN and Neural Network classifier is done using Weighted Ensemble to predict the number of map and number of reduce. The weighted ensemble to predict the number of maps is

$$NP = w_1 NP_{SVM} + w_2 NP_{KNN} + w_3 NP_{NN}$$

Where

NP_{SVM} is the prediction result from SVM classifier

NP_{KNN} is the prediction result from the KNN classifer

NP_{NN} is the prediction result from the Neural network classifier.

The weight values are calculated as

$$w_i = \log\frac{1 - error(C_i)}{error(C_i)}, 1 \le C_i \le 3$$

Where C_i is the corresponding classifier used for prediction of number of maps. Error of the classifier is calculated in terms of root mean square between the actual and the predicted number of maps. It is calculated as

$$error(C_i) = \sqrt{(NP_{C_i} - NP_{actual})}$$

Where

NP_{C_i} is the predicted value of number of maps by the classifier C_i and NP_{actual} is the actual value.

The Hadoop profiled data is split to 80:20 ratio. Each of the SVM, KNN and NN classifiers are trained to predict the number of maps. Testing is done with 20% test data to calculate the weight values.

The above procedure is repeated for finding the adaptive weighted model for number of reduce. It is given as

$$NR = w_1 NR_{SVM} + w_2 NR_{KNN} + w_3 NR_{NN}$$

Where

NR_{SVM} is the prediction result from SVM classifier

NR_{KNN} is the prediction result from the KNN classifer

NR_{NN} is the prediction result from the Neural network classifier.

V. NOVELTY IN THE PROPOSED SOLUTION

The overall architecture of the proposed solution is given in Figure 1 and the adaptive weighted ensemble model is given

 DOI : 10.35291/2454-9150.2021.0017

in Figure 2. Following are the important contribution in this work

1. Different from previous model based only on intermediate data, this work considers tasks specific computation and IO parameters, rate of change in intermediate data also in modeling the number of maps and number of reduce.
2. The solution is adaptive to different applications, as the weights are learnt dynamically based on past performance.
3. No assumption on linearity on intermediate data and relation is modeled using adaptive ensemble machine learning model.
4. The dynamics in intermediate volume is not considered in the previous work but the proposed solution considers it in prediction.

VI. RESULTS

The performance of the proposed solution is tested against PUMA dataset [18] in Hadoop environment. The performance of the proposed solution is compared for word count and k-means for different volume of datasets. The performance is compared with the solution proposed in [12] and default parameters of Hadoop. The solution [12] has proposed guidelines for selection of various configuration parameters.

The performance is compared in terms of Execution time.

The execution time is measured for different volume of the data and the result is given below

Data volume (MB)	Proposed ensemble	[12]	Default Hadoop
128	50	60	100
256	84	100	170
512	112	150	240
1024	202	270	430
2048	470	510	840

Table 3 :Execution time for different volume

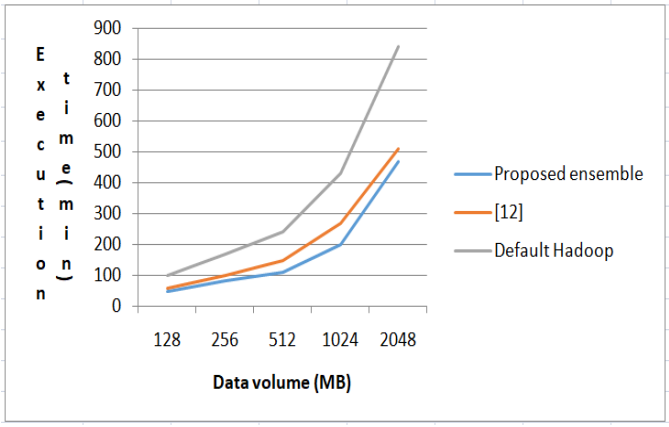

Figure 5 : Comparison of Execution time

The average execution time in the proposed solution is less than [12] by 19.12% and 94.53% compared to Default Hadoop. The results show that the proposed solution is above to provide a better value for number of maps and number of reduce compared to existing solutions.

The execution time is measured for different number of iterations for the same volume (128MB) of dataset and the result is given below.

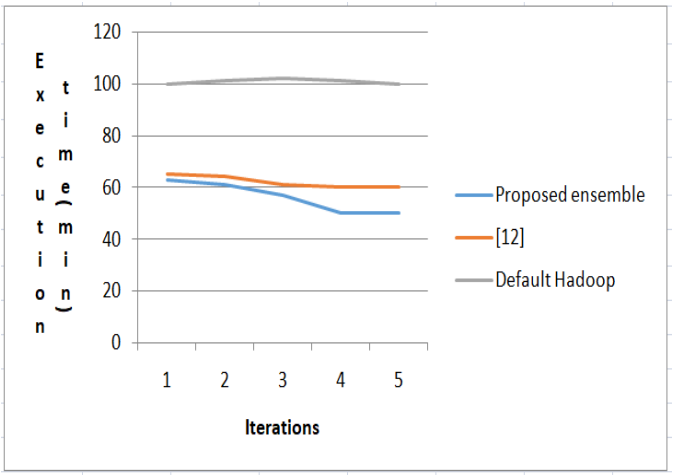

Figure 6 : Execution time over iterations

The execution time is learnt in adaptive manner in the proposed solution compared to [12]. Over the 5 iterations, the standard deviation in the proposed ensemble based solution is 6.05 minutes compared to standard deviation of 2.34 minutes in the solution [12]. The proposed solution has 61.3% more adaptability compared to [12].

The difference between the actual and prediction value is measured in terms of RMSE for different volume of dataset in the proposed ensemble model and the result is given below

Table 4 : RMSE in Proposed Solution

Data volume (MB)	RMSE for number of Maps	RMSE for number of reduce
128	1.73	2
256	3.16	3.74
512	4.6	4.7
1024	3.2	3.8
2048	4.4	4.3

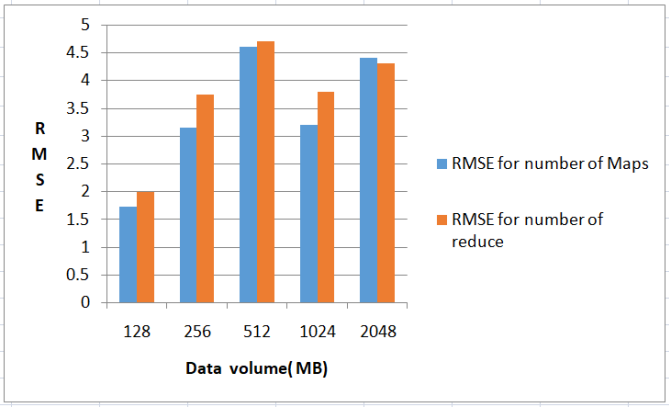

Figure 7 : RMSE for different data volume

RMSE value for number of map has a standard deviation of 1.153 and RMSE value of number of reduce has a standard deviation of 1.03. This implies irrespective of volume of data, the proposed model has good accuracy in predicting the

 DOI : 10.35291/2454-9150.2021.0017

number of maps and number of reduce. Also the average RMSE value for map is 3.4 and for reduce is 3.7, which means the prediction is very close to the actual value.

VI. CONCLUSION

An adaptive ensemble model for Hadoop parameter optimization is proposed in this work. The proposed model is based on multiple Hadoop execution parameters compared to most solution based on intermediate data volume alone. Different from previous solutions, this work also considers dynamics in intermediate volume. The weights for ensembling is adaptive to past performance predictions , thereby the proposed solution can adapt to different types of applications like CPU and IO intensive. Through performance results, the proposed solution is found to have 95.53% speedup compared to default Hadoop.

REFERENCES

[1] B. Nicolae, D. Moise, G. Antoniu, and al. BlobSeer: Bringing high throughput under heavy concurrency to Hadoop Map/Reduce applications. In Procs of the 24th IPDPS 2010, 2010. In press

[2] A. Verma, L. Cherkasova, and R. Campbell. Resource Provisioning Framework for MapReduce Jobs with Performance Goals. ACM/IFIP/USENIX Middleware, pages 165–186, 2011.

[3] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica. Improving mapreduce performance in heterogeneous environments. In USENIX Symposium on Operating Systems Design and Implementation (OSDI), volume 8, page 7, 2008

[4] Qi Zhang, "PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce", 2015 IEEE

[5] Zhenhua Guo , Geoffrey Fox , Mo Zhou, Investigation of Data Locality in MapReduce, Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), p.419-426, May 13-16, 2012 [doi>10.1109/CCGrid.2012.42]

[6] Adam Crume , Joe Buck , Carlos Maltzahn , Scott Brandt, Compressing Intermediate Keys between Mappers and Reducers in SciHadoop, Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, p.7-12, November 10-16, 2012 [doi>10.1109/SC.Companion.2012.12]

[7] Y. Chen, A. Ganapathi, and R. H. Katz, "To compress or not to compress-compute vs. io tradeoffs for mapreduce energy efficiency," in Proceedings of the first ACM SIGCOMM workshop on Green networking. ACM, 2010, pp. 23–28.

[8] W. Yu, Y. Wang, X. Que, and C. Xu, "Virtual shuffling for efficient data movement in mapreduce," IEEE Transactions on Computers, vol. 64, no. 2, pp. 556–568, 2015

[9] G. Ruan, H. Zhang, and B. Plale, "Exploiting mapreduce and data compression for data-intensive applications," in Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery. ACM, 2013, pp. 1–8

[10] D. Moise, T.-T.-L. Trieu, L. Boug´e, and G. Antoniu, "Optimizing intermediate data management in mapreduce computations," in Proceedings of the first international workshop on cloud computing platforms. ACM, 2011, pp. 1–7 .

[11] Veiga, Jorge & Expósito, Roberto & Taboada, Guillermo & Touriño, Juan. (2018). Enhancing in-memory efficiency for MapReduce-based data processing. Journal of Parallel and Distributed Computing. 120. 10.1016/j.jpdc.2018.04.001.

[12] Chen, Xiang & Liang, Yi & Li, Guang-Rui & Chen, Cheng & Liu, Si-Yu. (2017). Optimizing Performance of Hadoop with Parameter Tuning. ITM Web of Conferences. 12. 03040. 10.1051/itmconf/20171203040.

[13] Maria Malik, Hassan Ghasemzadeh, Tinoosh Mohsenin, Rosario Cammarota, Liang Zhao, Avesta Sasan, Houman Homayoun, and Setareh Rafatirad. 2019. ECoST: Energy-Efficient Co-Locating and Self-Tuning MapReduce Applications. In Proceedings of the 48th International Conference on Parallel Processing (ICPP 2019).

[14] C, K. and X, A. (2020), Task failure resilience technique for improving the performance of MapReduce in Hadoop. ETRI Journal, 42: 748-760. https://doi.org/10.4218/etrij.2018-0265

[15] Liao G., Datta K., Willke T.L. (2013) Gunther: Search-Based Auto-Tuning of MapReduce. In: Wolf F., Mohr B., an Mey D. (eds) Euro-Par 2013 Parallel Processing. Euro-Par 2013. Lecture Notes in Computer Science, vol 8097. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40047-6_42

[16] Bhaskar, Archana & Ranjan, Rajeev. (2019). Optimized memory model for hadoop map reduce framework. International Journal of Electrical and Computer Engineering (IJECE). 9. 4396. 10.11591/ijece.v9i5.pp4396-4407.

[17] S. Kumar, S. Padakandla, L. Chandrashekar, P. Parihar, K. Gopinath and S. Bhatnagar, "Scalable Performance Tuning of Hadoop MapReduce: A Noisy Gradient Approach," *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honolulu, CA, 2017, pp. 375-382, doi: 10.1109/CLOUD.2017.55

[18] https://engineering.purdue.edu/~puma/datasets.htm

 DOI : 10.35291/2454-9150.2021.0017