Replication Effect over Hadoop MapReduce Performance using Regression Analysis

Aisha Shabbir
School of Computing
Faculty of Engineering
University Technology Malaysia
Skudai, Johor Malaysia

Kamalrulnizam Abu Bakar
School of Computing
Faculty of Engineering
University Technology Malaysia
Skudai, Johor Malaysia

Raja Zahilah Raja Mohd. Radzi
School of Computing
Faculty of Engineering
University Technology Malaysia
Skudai, Johor Malaysia

ABSTRACT
Hadoop MapReduce is the community accepted platform that deals with the gigantic data in an efficient and cost-effective manner. To cope up with ever growing datasets and shrinking time to analyze them, Hadoop MapReduce leveraged parallelize computations on large distributed clusters consisting of many machines. Careful consideration of the factors affecting the Hadoop MapReduce can enhance its performance. Many researches has been done for improving the total job execution time of MapReduce by optimizing different parameters. The replication factor is still unexplored for its effect on the MapReduce job completion time. This paper focuses on the evaluation of data replication factor on MapReduce job completion time using regression analysis. The performance of the Hadoop MapReduce job in terms of total job completion time is monitored experimentally by changing different values of replication. The evaluation results evidently shows the dependence of the job completion time on the replication factor. The dependence of total job completion time on the replication has been verified both analytically and experimentally.

General Terms
Big Data Processing, Distributed Computing Systems, Statistical/Machine Learning Techniques, and Performance Analysis.

Keywords
Hadoop MapReduce, Big Data, Regression Analysis, Data Replication, Job optimization

1. INTRODUCTION
The dramatic evolution of digital world results in surges of volumes of data. The International Data Corporation Gens and Predictions [1] claimed that there might be a chance to have 40 folds data growth from the year 2012 to 2020 and expected that it would double for the every two years interval. The main production sources of big data are social media like Facebook, twitter, emails, mobile applications and the migration of manual to automatic of almost every entity. The amount of information has burst out each time and thus need for invention of a new storage method. For handling huge volume storage, Big Data storage companies such as IBM, EMC Amazon utilizing the tools like Apache Drill, SAMOA, NoSQL, IKANOW, Hadoop MapReduce and Horton Works [2]. Efficient processing and analyzing huge volume and variety of data has become the major source of innovation for compute intensive and data-intensive applications.

Hadoop is an indispensable component of the big data. It is an open source platform that uses the MapReduce model as a backbone. It is designed for executing data-intensive distributed computing applications. Hadoop comprises three main sub frameworks: Hadoop Common, Hadoop Distributed File System (HDFS), and Hadoop MapReduce. Hadoop Common offers the utility functions, including remote procedure call (RPC) facilities and object serialization libraries that are leveraged by the HDFS and MapReduce frameworks. HDFS is an implementation of a distributed file system that is based on Google's distributed file system, named GFS (Google File System). MapReduce is initially established by Google and it is designed for processing big data by exploiting the parallelism among a cluster of machines. Such parallelization enables compute frameworks to cope with growth in datasets being faster than Moore's law. The real implementation of MapReduce for huge scale data sets usually takes place on more than one machine or on a number of machines [3-4]. There are many factors which can affect the Hadoop MapReduce performance.

To achieve a better performance, the careful consideration of the factors is needed. There are some factors explored by many researches for Hadoop MapReduce optimization. Some researchers focused on the scheduling techniques to improve the overall job execution time of MapReduce jobs [5-8]. Similarly, some researches focused on particular phase scheduling [9-11]. Some research has been done to improve the fault tolerance [12-13]. Some tried to focus on the reliability issues [14]. There are many factors contributing towards the job execution. This study is focusing on the analysis of the unexplored factor i.e. replication factor towards the overall job execution time. The basic reason of making the replication in the distributed file system to increase the reliability and making the system fault tolerant. If any of the task's data is lost or the job is killed by any of the reason. Then, the replicated copy has been sent to complete the job. Suitable selection of this parameter is important because the inappropriate selection of replication factor may deteriorates severely the Hadoop MapReduce performance. For huge sets of data processing, large amount of data will be replicated over the computational resources and large storage capacity will be needed and it leads to huge burden over the network also.

Regression analysis comes under both statistical and machine learning techniques. It is one of the mostly chosen technique by researchers for making the predictions [15]. Regression analysis has widely used also for the selection criteria of a factor on a particular task. To elaborate it further, the regression analysis predictive values can make the hypothesis wrong or right. This study is focusing on the analysis of the contribution of replication factor towards the overall job

execution time. Let us suppose, for replication of the data of a certain size by two can be generally assumed as that it will delay a job completion time by two times of its basic execution time. Thus, to make this illustration true or false regression analysis will be performed. This study evaluates the dependence of replication factor over the total job completion time of Hadoop MapReduce through regression analysis. The general view of the idea described is shown in Figure 1.

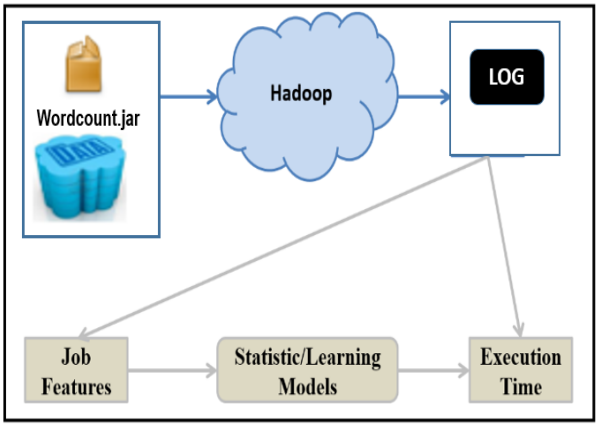

Figure 1: Computational framework overview

The organization of the paper is as follows: Section I is about the introduction. Section II describes the preliminaries i.e. giving the overview of Hadoop MapReduce and its architectural details. In addition, it briefly describes about the Regression analysis. Section III is about the problem insight. In Section IV, presents the experimental evaluation. Section V comprises the results and discussions. Last section i.e. VI is of conclusion and following then are the acknowledgements and references of this study.

2. PRELIMINARIES

2.1 An Overview of Hadoop MapReduce

The concept of the Hadoop was initiated by the google to work toward an innovative approach that has the capability of handling the enormous data. Later on, different procedures were introduced to deal with the storing, processing, and analysis of the avalanche of data. The Doug Cutting was among the pioneer and he used his son's toy name for this framework. It has the capability to deal with the gigantic data using its central characteristic i.e. the distributed system. Although the internal configuration of the Hadoop is complicated task, however, the detailed insight of the various component of the MapReduce job enable the developers and other associated personnel for required tuning according to the circumstances. The Hadoop is based on the Google File System and it uses the MapReduce that extends it processing from single server to thousands of servers. It provides a transparent parallel processing that uses many distributed nodes. It also provides transparency to the other features such as the reliability that hides the failure of the underlying tasks or subtasks, and the alliance of the processed results.

Hadoop MapReduce generally executes a job in two phases. During the map section, it divides the data input and run it on the given set of nodes. The mappers produces the output as a key and value pairs. These pairs are passed to the reducers to reduce it for the final result. Thus, in the reduce phase, the output of mappers are treated as input generally termed as intermediate data. There exists a merge and sort section named as shuffle between the map and reduce phase. In this shuffle phase, the data added to the mappers are divided and exchanged to the ideal machines executing the reduce section services.

2.2 Hadoop MapReduce Architecture

In the basic Hadoop MapReduce architecture, the master node runs two Hadoop components which are often called Hadoop daemons: NameNode and Job Tracker. Each slave node in the Hadoop cluster also runs two Hadoop daemons: DataNode and Task Tracker [16]. The NameNode and DataNode are the Hadoop daemons in charge of managing HDFS. Each file that is written to HDFS is split into blocks of 64 MB or 128MB and each block is stored on the storage device of the node where DataNode is running. In addition, each block is replicated three times (default value) and stored on different DataNode to provide data redundancy and availability. It is the job of NameNode to keep track of which DataNode stores the blocks of a particular file (which is referred to as the metadata of HDFS). Another important function of NameNode (master) is to direct DataNode (slaves) to perform HDFS block operations (creation, deletion, and replication). DataNode keep in constant contact with NameNode to receive instructions and also have to handle read and write requests from HDFS clients. Figure 2 gives an overview of the Hadoop MapReduce architecture.

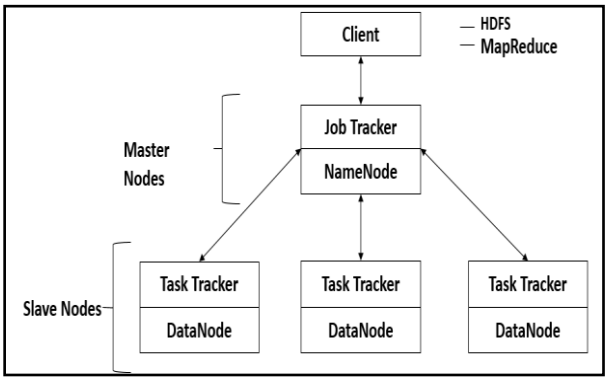

Figure 2: Hadoop MapReduce architecture [16]

2.3 Regression Analysis

Regression analysis comes under both statistical and machine learning techniques. Regression analysis has been used for the qualification and disqualification of a variable for the particular dependent variable or task. There are many configuration parameters for the Hadoop MapReduce job. One of the configuration parameter is the replication factor. Whether Hadoop MapReduce job performance really depends upon the replication or not will be explored using regression analysis in this research study. The qualification for dependence of the variable depends upon the prediction value of the regression analysis results. Predictive values normally called as P-values. If a P-value against the variable is less than 0.05 then the variable has no effect over the given task i.e. on the dependent variable and can be neglected in the overall calculation [17].

3. PROBLEM INSIGHT

Nowadays, Hadoop has been used typically in conjunction with cloud computing, for executing various Big Data applications, including web analytics applications, scientific applications, data mining applications, and enterprise data-processing applications [18]. A set of machines (where each machine is called a node) that runs a job by using the

MapReduce framework workflow is referred to as a MapReduce based cluster. MapReduce is proposed by Google to simplify massively distributed parallel processing so that very large and complex datasets can be processed and analyzed efficiently.

The MapReduce system carried out the job execution in Hadoop and Hadoop Distributed File System (HDFS) such that the input data is loaded and subdivided into pieces, with each piece of data is replicated over multiple nodes. The default value of the replication is three. When a user uploads files to HDFS, files could be splitted into chunks with the range of a chunk size from 64MB to 128MB. Thus, on submission of a job, the input data is divided according to the split size defined by the user. Thus, for the job size bigger than a fundament split size requires more than one node. MapReduce accomplish job processing on huge datasets by supposing that large dataset storage is distributed over a large number of machines. The computation is done in two main phases i.e. Map and Reduce. There is another phase i.e. shuffle phase also for dealing with intermediate data. The details of the MapReduce job computation are as follows:

i. Preparation of the input data: MapReduce uses Google's file system which is called GFS or HDFS, as an underlying storage layer for reading input and saving output. GFS is a distribution system distributed block that supports error tolerance by separating and replicating data.

ii. Map phase: The input data is divided into small chunk on different worker nodes and results are in form of key value pairs.

iii. Shuffle phase: The output from mappers is sorted and sent to reduce processors.

iv. Reduce phase: The sorted output data group (each key) is processed in parallel to each reduced node. User-defined reducing functions are implemented once for every key value generated by map steps.

v. Final output: The MapReduce system finally collects all of the reduce outputs to produce the final output. The results are stored in the GFS.

MapReduce computes a job into different phases and during its operation it follow the general workflow as shown Figure 3 for all types of jobs.

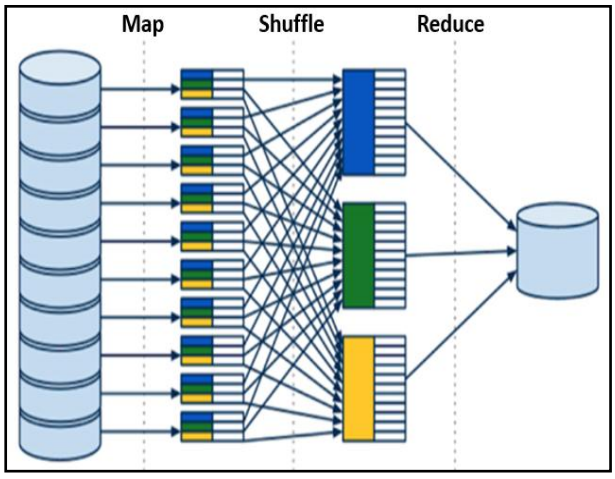

Figure 3: MapReduce workflow [19]

4. EXPERIMENTAL EVALUATION

The implementation and evaluation of this work has been conducted in Hadoop version 2.7.1. Several MapReduce jobs with different sizes of input data ranging from Bytes to Giga bytes has been evaluated against different replication figures. Hadoop MapReduce job has many configuration factors including replication factor. The configuration settings with the replication value 1 is shown in the Figures 4. Regression analysis has been done in the Microsoft excel. The total job completion time has been used as a performance evaluation metric.

name	value
dfs.replication	1
dfs.namenode.name.dir	file:/Syncfusion/BigData/3.2.0.20/BigDataSDK/Metadata/data/dfs/namenode
dfs.datanode.data.dir	file:/Syncfusion/BigData/3.2.0.20/BigDataSDK/Metadata/data/dfs/datanode
dfs.permissions	false
dfs.http.address	localhost:50070
dfs.webhdfs.enabled	true
dfs.datanode.du.reserved	1073741824
dfs.datanode.data.dir.perm	777
dfs.namenode.rpc-bind-host	0.0.0.0

Figure 4: MapReduce job configuration with replication 1

5. RESULTS AND DISCUSSION

It has been observed that the variation of the replication factor has a great effect on the total job completion time. The results are extracted from the logs after running the Hadoop MapReduce Jobs. The extraction is done for the total job execution time against different values of replication factor and are given in the Tables 1-3. The replication factor has been changed from 1 to 5. The replication values in the configuration settings can be viewed from Figures 4. The input data size has been varied from Bytes to Gigabytes (GB) for each replication value. It has been observed that changing the input data size has effect on total job completion time. Figure 5 is clearly depicting that with the increase in input data size the total job completion time also increases. Figure 5 also shows that with the increase of replication value over the same sets of data the total job execution time becomes more. The default value of the replication factor is 3 for the Hadoop MapReduce. Moreover, with increasing the input data size with even default values results in much larger total execution time.

The core idea of making the replication is to enhance the reliability. On the other hand, replication of big data sets results in wastage of resources as well. Thus it is important to consider the cases where replication is necessary and where it deteriorates the system's performance. Several simulations has been done and simulation results are analyzed. Hadoop, the open source implementation of MapReduce runs in three modes standalone, pseudo-distributed and fully distributed. Figure 6 results shows the long list of job history against the replication factor with the minimum value i.e. 1. More than hundred simulations with different configuration in standalone and pseudo distributed modes has been done. The point which is noticeable that all the jobs are successfully completed with all the tasks of every job as depicted from Figure 8. Thus, doing the replication in standalone and pseudo-distributed modes has no point of failure as observed by the simulation results. Unless if there will be some hardware failure occurs.

Thus the replication for these cases would merely increase the total job execution time and results in in-efficient usage of storage assets.

Several simulations has been done and it is difficult to present all the simulation results here. The total job execution time results against different input data sizes from some bytes to Gigabytes with replication factor as 1 has been shown in the Table 1.

Table 1: Job completion times with replication 1

Input data size (MB)	Replication	Total Execution Time (sec)
0.0005	1	20
120.42	1	38
260.76	1	43
521.53	1	50
1180	1	101
1600	1	121
2005	1	143
2600	1	178
3050	1	205
3500	1	217
3900	1	257
4155	1	266
4700	1	310
5100	1	380
5700	1	399
6110	1	408

The total job execution time results against different input data sizes from some bytes to Gigabytes with replication factor as three has been shown in the Table 2.

Table 2: Job completion times with replication 3

Input data size (MB)	Replication	Total Execution Time (sec)
0.0005	3	19
120.42	3	39
260.76	3	43
521.53	3	67
1180	3	159
1600	3	165
2005	3	171
2600	3	178
3050	3	183
3500	3	242
3900	3	341
4155	3	480
4700	3	550
5100	3	589
5700	3	623
6110	3	729

The total job execution time results against different input data sizes from some bytes to Gigabytes with replication factor as five has been shown in the Table 3.

Table 3: Job completion times with replication 5

Input data size (MB)	Replication	Total Execution Time (sec)
0.0005	5	20
120.42	5	39
260.76	5	43
521.53	5	67
1180	5	155
1600	5	181
2005	5	239
2600	5	298
3050	5	361
3500	5	410
3900	5	492
4155	5	503
4700	5	590
5100	5	680
5700	5	710
6110	5	757

For analytical analysis, the technique widely adopted by the research community i.e. regression analysis has been used. Regression analysis has been used for predictive analysis and for the qualification of a variable dependence.

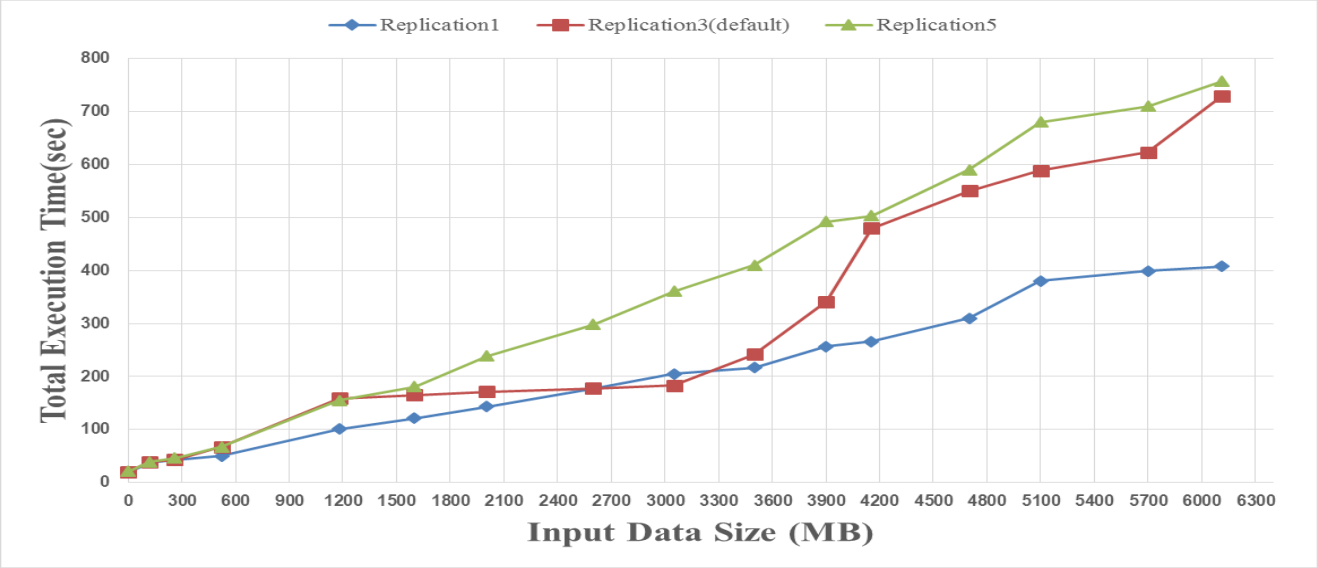

Figure 5: Total job completion time vs different input data size

Search:

Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
job_1537367648356_0007	word count	SYSTEM	default	SUCCEEDED	10	10	1	1
job_1537367648356_0006	word count	SYSTEM	default	SUCCEEDED	4	4	1	1
job_1537367648356_0005	word count	SYSTEM	default	SUCCEEDED	2	2	1	1
job_1537367648356_0004	word count	SYSTEM	default	SUCCEEDED	25	25	1	1
job_1537367648356_0003	word count	SYSTEM	default	SUCCEEDED	49	49	1	1
job_1537367648356_0002	word count	SYSTEM	default	SUCCEEDED	1	1	1	1
job_1537367648356_0001	word count	SYSTEM	default	SUCCEEDED	1	1	1	1
job_1537267166660_0023	word count	SYSTEM	default	SUCCEEDED	123	123	1	1
job_1537267166660_0022	word count	SYSTEM	default	SUCCEEDED	49	49	1	1
job_1537267166660_0021	word count	SYSTEM	default	SUCCEEDED	33	33	1	1
job_1537267166660_0020	word count	SYSTEM	default	SUCCEEDED	25	25	1	1
job_1537267166660_0019	word count	SYSTEM	default	SUCCEEDED	17	17	1	1
job_1537267166660_0018	word count	SYSTEM	default	SUCCEEDED	15	15	1	1
job_1537267166660_0017	word count	SYSTEM	default	SUCCEEDED	10	10	1	1
job_1537267166660_0016	word count	SYSTEM	default	SUCCEEDED	4	4	1	1
job_1537267166660_0015	word count	SYSTEM	default	SUCCEEDED	2	2	1	1

Figure 6: Different jobs successful completion along with all the subtask

Table 4: Regression analysis results

	Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%
Intercept	-8.161713495	37.5138	-0.21757	0.830215	-86.9753	70.65185	-86.9753	70.65185
Replication	7.25	10.30406	0.703606	0.490681	-14.398	28.89804	-14.398	28.89804

The regression analysis has been done over the simulations results to verify the effect of replication on total execution time. The results of regression analysis are shown in Table 4. The prediction value i.e. the P-value of the regression has been used for the qualification criteria for a factor. If the p-value against a specific factor is less than 0.05 then the factor has no significant contribution towards the testing experiment and that factor can be ignored even in analysis. The P-value for regression factor as shown in Table 4 is 0.490681 which is greater than 0.05. Thus, this P-value of replication among regression results enables the qualification of replication factor on total execution time.

6. CONCLUSION

Hadoop MapReduce platform is considered as the most prominent and effective for the big data problems that allow the processing of gigantic data over many underlying distributed nodes. A MapReduce job has many configuration parameters and their optimal tuning can significantly increase the Hadoop MapReduce performance. In this paper, an unexplored factor i.e. the replication factor is evaluated for its effect on the Hadoop MapReduce total job completion time. Several simulations has been done by different replication values and also by varying the input data sizes. It has been observed that the variation of the replication factor influences the total job completion time. The results from the Figure 5 shows that increasing the input size with constant replication has completed a job execution in longer time. Furthermore, an increase in the replication value will results in higher values of total job completion time. In addition, Regression analysis has been used for qualification criteria of a variable on the dependent one. The evaluation results evidently shows the dependence of the job completion time on the replication factor. The results are also validated through regression analysis p-value. From the Table 4 we can see that p-value is greater than 0.05 which means replication factor contributes as major to job completion time. Consequently, the replication factor has a substantial effect on overall performance of the Hadoop MapReduce.

7. ACKNOWLEDGMENTS

Special thankful to University Teknologi Malaysia (UTM) for providing the good research environment, tools and facilities to accomplish this research work. I would like to express my very great appreciation to my supervisor Prof. Dr. Kamalrulnizam Abu Bakar and co-supervisor Dr. Raja Zahilah Raja Mohd. Radzi for their guidance and professional support for the research. I am thankful and indebted to my senior Tasneem Darwish for her timely contributions and guidance.

8. REFERENCES

[1] Gens,F., and Predictions, I. (2015)Team IDC Predictions.

[2] Agneeswaran, V. S. (2014). Big data analytics beyond hadoop: real-time applications with storm, spark, and more hadoop alternatives: FT Press.

[3] Delimitrou, C., and Kozyrakis, C. (2014). Quasar: resource-efficient and QoS-aware cluster management. Paper presented at the ACM SIGPLAN Notices,127-144.

[4] Prajapati, V. (2013). Big data analytics with R and Hadoop: Packt Publishing Ltd.

[5] Balagoni, Y., and Rao, R. R. (2017). Locality-Load-Prediction Aware Multi-Objective Task Scheduling in the Heterogeneous Cloud Environment. Indian Journal of Science and Technology, 10(9).

[6] Althebyan, Q., Jararweh, Y., Yaseen, Q., AlQudah, O., and Al-Ayyoub, M. (2015). Evaluating map reduce tasks scheduling algorithms over cloud computing infrastructure. Concurrency and Computation: Practice and Experience, 27(18), 5686-5699.

[7] Chen, Q., Zhang, D., Guo, M., Deng, Q., and Guo, S. (2010). Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. Paper presented at the Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on, 2736-2743.

[8] Tang, Z., Liu, M., Ammar, A., Li, K., and Li, K. (2016). An optimized MapReduce workflow scheduling algorithm for heterogeneous computing. The Journal of Supercomputing, 72(6), 2059-2079.

[9] Ke, H., Li, P., Guo, S., and Guo, M. (2016). On traffic-aware partition and aggregation in mapreduce for big data applications. IEEE Transactions on Parallel and Distributed Systems, 27(3), 818-828.

[10] Tiwari, N., Sarkar, S., Bellur, U., and Indrawan, M. (2015). Classification framework of MapReduce scheduling algorithms. ACM Computing Surveys (CSUR), 47(3), 49.

[11] Neelakandan, S., Divyabharathi, S., Rahini, S., and Vijayalakshmi, G. (2016). Large scale optimization to minimize network traffic using MapReduce in big data applications, 2016 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC), 193-199.

[12] Fu, Huansong, Haiquan Chen, Yue Zhu, and Weikuan Yu."Farms: Efficient Mapreduce Speculation for Failure Recovery in Short Jobs."Parallel Computing (2017): 68.

[13] Xu, Huanle and Wing Cheong Lau. "Optimization for Speculative Execution in Big Data Processing Clusters." IEEE Transactions on Parallel and Distributed Systems 28, no. 2 (2017): 530-45.

[14] Yan, W., Xue, Y., and Malin, B. (2013). Scalable and robust key group size estimation for reducer load balancing in MapReduce. Paper presented at the Big Data, 2013 IEEE International Conference on, 156-162.

[15] M. Khan, Yong Jin "Hadoop Performance Modeling for Job Estimation and Resource Provisioning", IEEE Transactions On Parallel And Distributed Systems, Vol. 27, No. 2, February 2016.

[16] Li, K.-C., Jiang, H., and Zomaya, A. Y. (2017). Big Data Management and Processing: CRC Press.

[17] http://www.statisticssolutions.com/directory-of-statistical-analyses-regression-analysis/regression/

[18] Bechini, A., Marcelloni, F., and Segatori, A. (2016). A MapReduce solution for associative classification of big data. Information Sciences, 332, 33-55.

[19] Li, R., Hu, H., Li, H., Wu, Y., and Yang, J. (2016). MapReduce parallel programming model: a state-of-the-art survey. International Journal of Parallel Programming, 44(4), 832-866.