Designing a Hadoop MapReduce Performance Model using Micro Benchmarking Approach

Manal Alalawi[1] and Herbert Daly[2]
[1,2]Institute of Research and Applicable Computing (IRAC), University of Bedfordshire, Luton, UK

ABSTRACT

Hadoop MapReduce platforms are currently used at an extensive rate to deal with complex data analysis of large size data sets. In MapReduce environments, parallel and distributed processing of big data is done with high energy requirements. Many of the contemporary organisations are looking to reduce the energy requirements of Hadoop MapReduce by maintaining the same performance levels. In this paper a platform performance model is proposed specifically for Hadoop MapReduce environments in order to improve the energy efficiency of these applications in automatic manner. Unlike the existing performance models, the proposed performance model related the different number of processed data and durations of executed phases that are accomplished through collected measurements from executed sets of micro benchmarking. The resource distribution strategy of this performance model helps to estimate the job completion time on the basis of resource distribution. Mathematical modelling and experiments showed the accuracy of this performance model is improving the energy efficiency of Hadoop MapReduce environments.

Keywords: Hadoop, MapReduce, energy efficiency, performance model, MapReduce phases

Introduction

Big data has brought the paradigm shift in both academia and industry. Most of the organizations based on big data are progressively using Hadoop with MapReduce model, which is an open-source framework to execute complex data analysis and to process large data sets (Jeffrey Dean, 2008). Apache Hadoop have several remarkable features to process the Big Data by implementing a number of different components to communicate with each other across multi node system. Hadoop MapReduce is based on large and complex framework, therefore the performance of Hadoop is highly important to execute the different jobs on each component such as network infrastructure, hardware and MapReduce configuration parameters which are more 190 (Babu, 2010).

In house Hadoop based clusters are facing a key challenge to have an automatic mechanism to process data intensive applications and to manage the resources distributions (White, 2015). There are limitations with default Hadoop MapReduce implementation that cannot provide resource management support for latency-sensitive applications and make the user to be responsible to figure out the required amount of resources for running jobs which affects the Hadoop performance. Additionally, MapReduce frameworks with different configuration parameters have significant effects on Hadoop job's performance. These distributed systems are developed on commodity hardware, therefore, the hardware heterogeneity, and the non-deterministic behavior of various applications increase the job execution time which directly affect the energy efficiency of Hadoop cluster (Jeffrey, 2011).

To make energy efficient Hadoop based cluster, most of the researchers consider the Hadoop parameter setting to increase the system performance as it plays a critical role for execution and also as these parameters have a significant impact on Hadoop system. It is time consuming process to tune the optimum values of these complex parameters manually which make the MapReduce framework as a black box (Jacob, 2010). It is exceptionally difficult to present an objective function or to make a mathematical model which can give a correlation between these parameters. Manually tuning of this complex system has become harder when the data size is increasing together with the difficult correlations between the configurations parameters. Therefore, it is highly important to have an effective and automatic approach to increase the energy efficiency of Hadoop parameters (Morton, 2010).

The paper intends in proposing and designing a performance model framework for Apache Hadoop MapReduce to make it more energy efficient by doing experiments on virtual environment. This performance model consists of resource distribution strategy which helps to estimate the job completion time on the basis of resource distribution. This approach works with data intensive complex applications that have concurrent and sequential jobs.

The reminder of this paper is organized as follows, Hadoop MapReduce framework is detailed in section II, and the state-of-art works in the literature of performance model for Hadoop framework are given in section III. Section IV presents design of Hadoop MapReduce performance model with evaluation of its accuracy and Section V summarizes the key findings of this report.

Hadoop MapReduce framework

In a MapReduce model, computations are expressed in terms of two functions: map and reduce that are user-defined. The map function performs the computation on each record and reduce function combines the output to give the answer after the counting, shuffling, sorting, and merging operations. This MapReduce programming model presents the simple interfaces for users to compute different operations by using both functions. The map function executes on each Input key-value pair record and produces the output as an intermediate key value pair data, the reduce function work on intermediate data and give the results. The Hadoop MapReduce parallelization and coordination process is given in Figure 1.

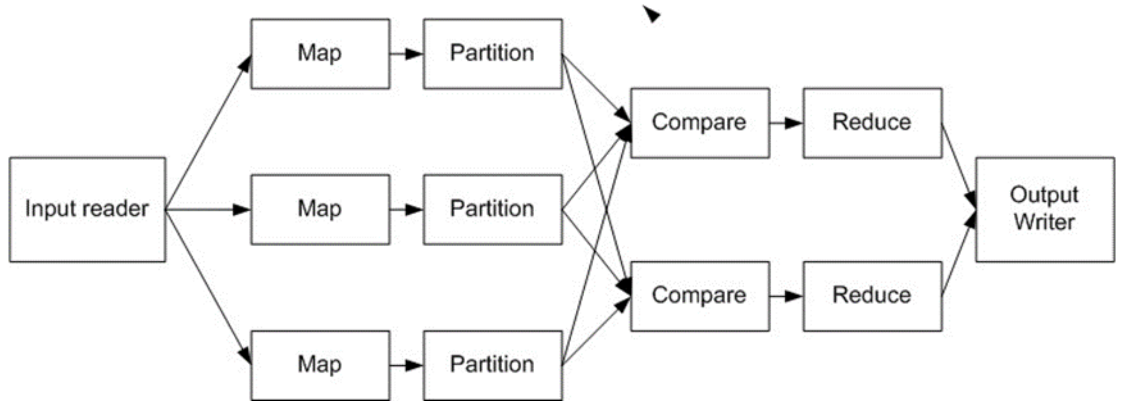

Figure 1 Hadoop MapReduce Computation flow diagram (Bryant, 2015)

A MapReduce execution step to store big data in distributed cluster depends on user configuration. The input file split into 16 to 64MB is distributed on to the cluster. In this cluster, one machine executes as a Master and other works as a Slave, also known as Workers. The Master assigns map tasks and Salve assigns reduce tasks. The intermediate key- value pair data is generated from the recode when worker executes the Map function and saved in memory (Bu, 2010).

The remote procedure call, running on the idle reducer workers gets the information about the data and partitions settings from Master. After getting all the information about intermediate data, reducer worker performs shuffling tasks, which includes sorting and grouping to produce intermediate keys (Yingjie, 2014). After analyzing the intermediate keys, the data analyzed by reducer workers and all the values is passed to the reducer function and output file is appended. Reduce file is the final output with combined data of intermediate keys and it is frequently used as feedback for subsequent map reduce or any other distributed file application.

Literature Review

Hadoop MapReduce performance model is an emerging research area that considersvarious activities such as scheduling, job optimization, and resource provisioning and job estimation. In the literature, different researchers proposed a number of performance models to make the Hadoop based system more energy efficient. Morton (2010) proposed performance models namely Parallax and an enhanced model called ParaTimer to estimate the Hadoop performance by executing the Pig queries and translating these queries into sequence of MapReduce jobs. To predict the progress of map and reduce phases, they have used the same query for input and output data to run the debug process. To predict the performance of Hive in Hadoop system, Ganapathi (2012) implemented the regression technique called Kernel Canonical Correlation Analysis (KCCA). The study by Ganapathi (2012) has demonstrated the particular tasks of KCCA technique that can be applied for MapReduce framework.

By using the Machine Learning (ML) techniques, Kadirvel (Fortes, 2012) proposed a performance model to predict the performance levels of Hadoop MapReduce jobs. Even though, this performance model analysed the performance of Hadoop MapReduce implementations, there is a lack of mathematical model to estimate the job execution. Lin (2012) proposed a cost vector to estimate the execution time of different map and reduce tasks. This cost vector model consists of computational complexity, disk I/O expenditures, network traffic, internal sort and CPU usage for estimating this execution time, even though it is very difficult and challenging job in Hadoop based cluster to accurately estimate the job execution time for multi tasks situation because there is competition for resource utilization in between the tasks. Additionally, in this cost vector performance model, Lin (2012) has presented the job execution time for map tasks and ignored the execution time for reduce tasks. Furthermore, Cui and Lin (2013) employed a simulator to evaluate the performance model effectiveness in a situation of task failure. Unlike the analytical approaches, the simulator based approaches give errors for the complex applications like MapReduce. Therefore, it's difficult to design an accurate simulator that can systematically simulate the complex jobs and understand the dynamics of MapReduce applications (Zhenhua, 2012).

Hadoop MapReduce Performance Model

In this section, we describe the Hadoop MapReduce Performance Model that can be used to estimate and improve the energy efficiency of Hadoop MapReduce implementations. This performance model is a relation between different number of processed data and durations of executed phases that are accomplished through collected measurements from executed sets of micro benchmarking. Therefore, executed phases and job completion time as a function of processed data are considered in this Hadoop MapReduce Performance Model.

Let's explain how to develop this platform performance Model as$Model_{\&'()*)}$. We have constructed six sub-models *H-M$_1$, H-M$_2$, H-M$_3$, HM$_4$, H-M$_5$*and *H-M$_6$*that defines the relationships for *Read, Collect, Spill, Merge, Shuffle and Write* respectively of a given Hadoop cluster. We use a Collected Platform Profile illustrated in figure to derive these sub-models.

Let Hadoop-Mi be a sub-model having solution (A$_i$, B$_i$) where i=1, 2, 3, 4, 5, 6. If the platform profile has k rows then$Data_{.}^{/}$and $Duration_{.}^{/}$are respectively the amount of data and duration in row j of this profile, where j=1, 2, 3… k. From linear regression method we have the following sets of equations:

$$A_{.} + \quad B_{.} \quad .\, Data_{.}^{/} = \quad Duration_{.}^{/} \text{ - } \quad (1)$$

Hadoop-M$_i$=(A$_i$, B$_i$). (2)

Where i=1, 2,..., 6 and j=1, 2, 3,…, k

The set of equations may be solved by different techniques and methods, so we may proceed to have a solution for set of equations (2) with least squares regression method as it minimizes the risk of absolute errors. Linear regression is strictly advised as it maintains overall accuracy and lessens the effects of bad measurements if taken.

The relation of $A_{.} + B_{.}$ in equation (1) and (2) gives a working sub-model as function of processed data during the execution of different phases i, and hence $H - M_{.} = (A_{.} + B_{.})$. So, Hadoop MapReduce performance model:

$$Model_{\&'()*} = (H - M_{>}, H - M_{@}, H - M_{A}, H - M_{B}, H - M_{C}, H - M_{D}) - (3)$$

In the experiments, a random phase will be expected to have a distinctive or exclusive behaviour while processing data of amount more or less than around 64.0 GB. For a good result, we use linear function that is comprised by two segments, first one to process a dataset up to 64.0 GB and second one to process dataset larger than 64.0 GB. The proposed platform performance model for Hadoop MapReduce is derived by execution of a set of micro-benchmarks on a small five node clusters and their respective phase durations. Each machine consists of two Intel processors, 8GB RAM and two 160GB hard disks. Hadoop 2.x is used with two machines referred as NameNode and JobTracker. Each of the working nodes is organized with 2 maps and 1 reduce slot. The block size of the file system is allocated for 64MB and its replication level is three. Speculative execution has no considerable role in set up therefore we set it disabled.

The interaction of processed data amount and varied phase execution timings for a given Hadoop cluster are depicted in Figures2-5. These figures reflect the platform profile for the phase's read, collect, spill and merge of the phases of the map task execution respectively. The Figures 6 and 7 depicts the shuffle and write phases in the reduce task respectively. Phase durations are scaled along Y-axis and processed data amounts are taken along X-axis, then the resulting graph are the scattered dots. A regression line is drawn along these dots in black colour, which represents the best fitted solution for the six phases. The shuffle phase in Figure 7 is expected to be approximated by a linear piecewise function that was already comprised by two further linear functions, and therefore a regression line gives a fair accuracy for this phase. For the authentication of shuffle phase a set of experiments has to be carried out. In these experiments, eachJava Virtual Machine (JVM) is configured with 2GB RAM and Hadoop limit is 46% of the allocated memory for in-memory buffer.

The segments of data after shuffling are mergesorted in memory and a spill file of size 1GB if new, is written to disk. After the shuffling of whole data, Hadoop merg-sorts initial ten spilled files are written in a new sorted file. Same procedure is carried out for the next ten files and so on till it ends. All the new sorted files are merg-sorted then. Thus we get prominently different shuffle performance to process intermediate data of amount larger than 10 GB. A linear piecewise function may check it performance more accurately.

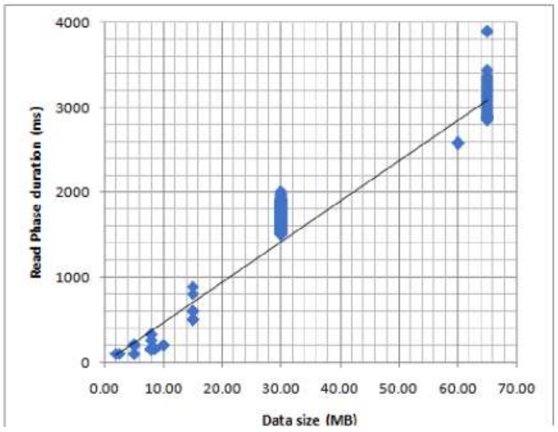

Figure 2 read phase

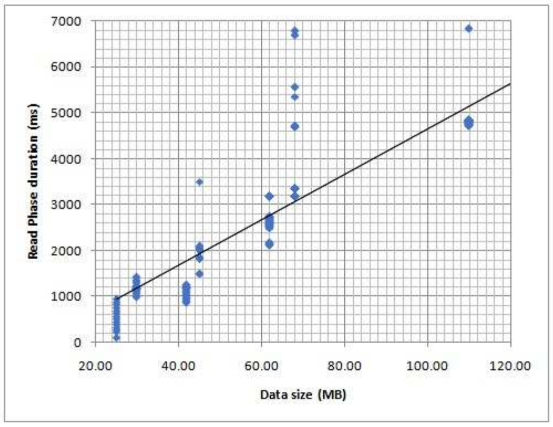

Figure 5 merge phase

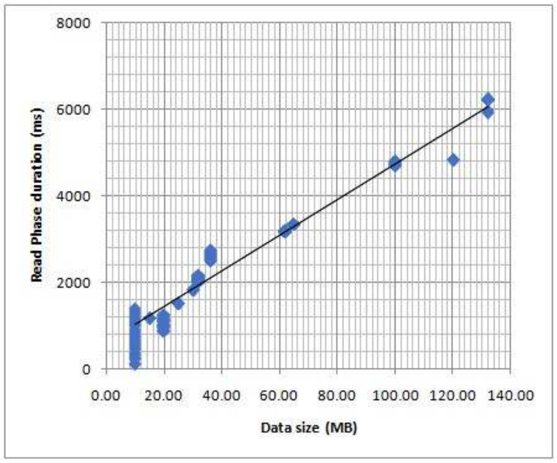

Figure 3 collect phase

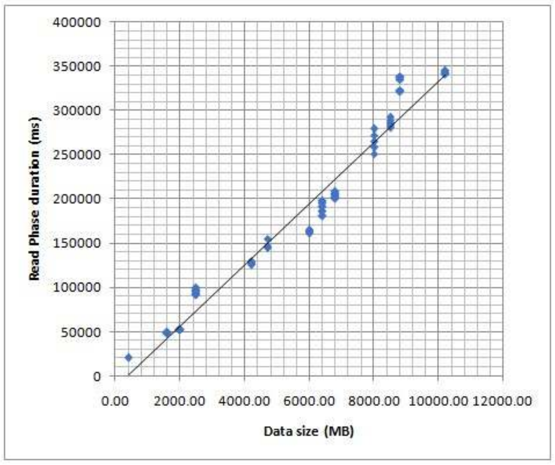

Figure 6 shuffle phase

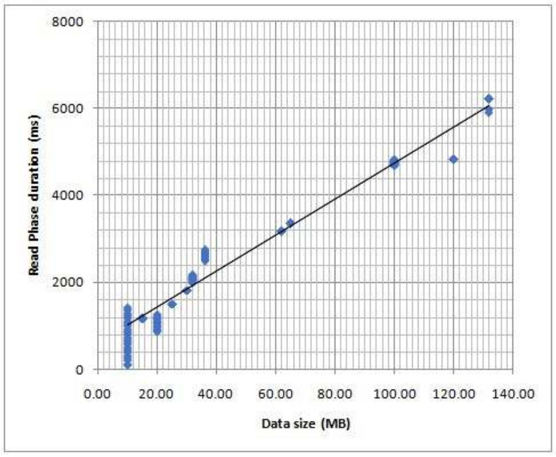

Figure 4 spill phase

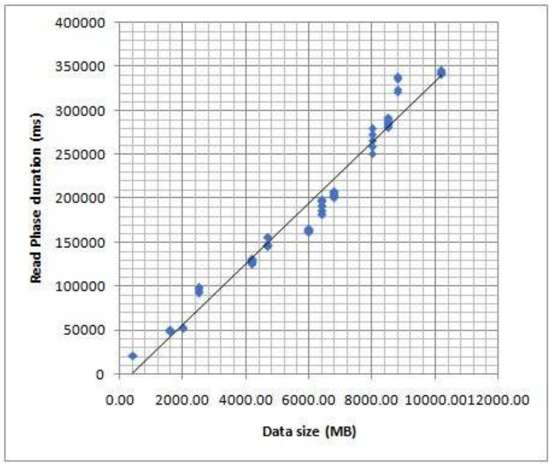

Figure 7 write phase

Accuracy of the MapReduce platform Performance Model

To evaluate the accuracy and precision of the proposed Hadoop MapReduce performance model, $Model$&'()*), the expected error that can be calculated for each data point in dataset is estimated. Let $duration.$&EFis a predicted duration and $duration.$GEFis a measured duration in row jof the platform profile against phasei and Mi is the function of Dataj. Now a relative error can be calculated as:

$error. =$ | FIE(J.KLFIE(J.KLM^{NOP} Q FIE(J.KLMNOP M^{ROP} - (4)

For each data point, relative errors are computed in platform profile. Cumulative Distribution Function (CDF) of relative errors is shown in Figure 8 for all the six phases considered above. CDF of relative errors shows that this performance model fits well to this data in experiment. Relative errors in derived models are shown in Table 1for all the six different processing phases.

Phase “Read”: During the ‘read’ phase less than 10 percent relative errors is found in 66 percent of map-tasks whereas less than 20 percent relative errors are found in 92 percent of map-tasks.

Phase “Shuffle”: During the ‘shuffle’ phase less than 10 percent relative errors is in 76 percent of reduce-tasks whereas less than 20 percent relative errors are found in 96 percent of reduce-tasks.

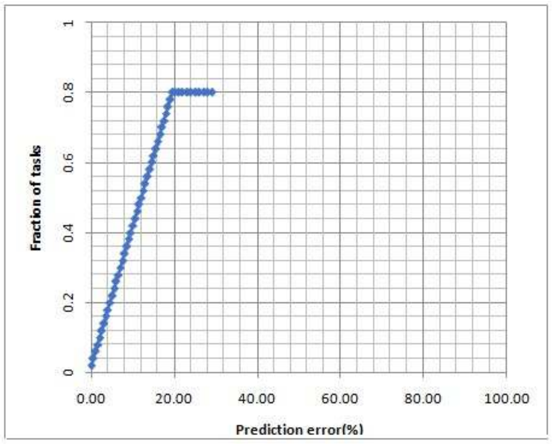

Figure 8 Cumulative distribution function of prediction error

MapReduce Phases	Error ≤ 10%	Error ≤ 20%	Error ≤ 30%
read	59%	75%	74%
collect	54%	67%	90%
spill	59%	67%	92%
merge	55%	78%	91%
shuffle	68%	79%	88%
Write	89%	87%	98%

Table 1: Relative errors in different phases

To validate the accuracy of developed Hadoop MapReduce performance model, various TeraSort and WordCount applications are implemented to calculate the execution time for different phases in a Hadoop based cluster. These applications are run on same five-node cluster and comparison is made between measured and predicted phase duration as discussed below. Based on the developed performance model, WordCount and TeraSort applications are executed on a 5 node cluster and comparison is done between the predicted phase execution time and measured phase duration. By using the TeraGen program, data of size 2.5 GB is generated for both the applications. Experiments are executed 5 times and averages of resulting measurements for the six generic phases are made and shown in Figures9 and 10. The resulting measurements of five experiments are expressed in Figures 9 and 10 by making a comparison between predicted and measured timings. In both the executed job applications, the number of reduce tasks is set to 50. The graph points in Figure 10 that built up the performance model can give each phase timings accurately. In Figure 9, the difference between predicted and measured duration ranges to 10 percent mostly, except for the shuffle phase which is near 18 percent specifically for WordCount Application.

In order to assess if this performance model designed for small test cluster is workable for larger ones, all the test jobs are repeated for the large cluster having better configuration and extra machines.

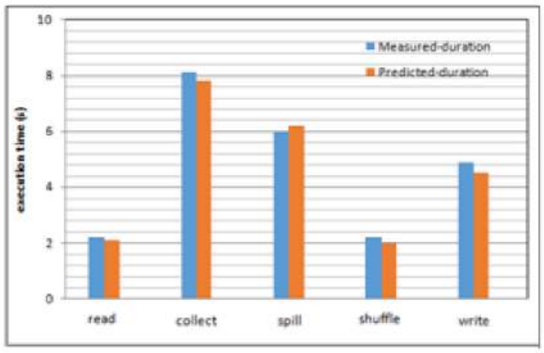

Figure 9 Word Count

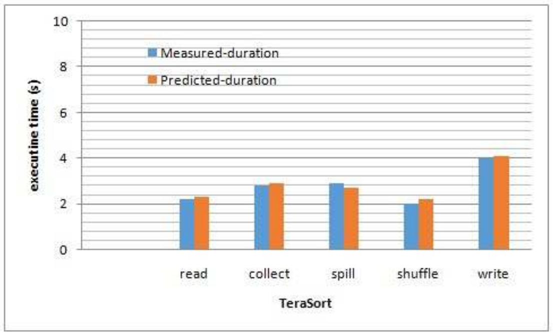

Figure 10 validating the accuracy of the MapReduce performance model on virtual 5-nodet cluster

The machines having same hardware as that in small test cluster are adjusted in 2 racks and interconnected with a GB Ethernet. In this cluster, two machines are taken as NameNode and JobTracker and Hadoop 2.0 is used while other 5 machines are set as workers. One reduce slot and two maps are provided in each work machine and the reduce tasks are set to 60 in each application. Figure 10 provides the measured duration and predicted durations for all the five phases. Critical analysis of the measured and predicted durations for all the five phases indicates that there exists very small difference between these durations. Hence the results validate the effort of constructing of platform performance model with the help of small test cluster.

Conclusion

Energy efficient operation of Hadoop MapReduce clusters has become highly crucial in order to minimise the energy costs of these clusters during data processing. Identifying the increased energy costs of the Hadoop MapReduce applications, this paper has designed a platform performance model suitable for Hadoop MapReduce clusters. Unlike the existing performance models, the proposed performance model related the different number of processed data and durations of executed phases that are accomplished through collected measurements from executed sets of micro benchmarking. The resource distribution strategy of this performance model helps to estimate the job completion time on the basis of resource distribution. The analytical model and experiments of this performance model are carried out with respect to five phases such as read, write, collect, spill and shuffle. The accuracy of this proposed performance model is also validated on both the small test and large test cluster. Improving this performance model by adding work flow and job execution features forms an interesting part of the future work.

References

Workshops (CLUSTER WORKSHOPS)

(pp. 231–239). IEEE.

• Xuelian Lin, Z. M. (2012). A Practical

• Fortes, S. K. (2012). Grey-Box Approach Performance Model for Hadoop for Performance Prediction in Map- MapReduce . IEEE International Reduce Based Platforms. Computer Conference on Cluster Computing Communications and Networks (ICCCN), Workshops (pp. 231 - 239). Beijing: IEEE.

2012 21st International Conference (pp. • Yingjie Guoa, L. W. (2014). The

1–9). IEEE. Improved Job Scheduling Algorithm of

• Ge Song, Z. M. (2011). A Hadoop Hadoop Platform.

MapReduce Method . High Performance Computing Performance Prediction • Yingyi Bu, B. H. (2010). HaLoop: and Communications & 2013 IEEE efficient iterative data processing International Conference on Embedded on large clusters . (pp. 285-296). and Ubiquitous Computing VLDB Endowment . (HPCC_EUC), 2013 IEEE 10th • Zhenhua Guo, G. F. (2012).

International Conference on, IEEE, pp.

820 - 825. Zhangjiajie. Improving Resource Utilization in

• J. Jeffrey Hanson. (2011). An introduction MapReduce. Indiana University, to the Hadoop Distributed File System . School of Informatics and Retrieved 2015, from IBM Developer Computing , Bloomington, IN.

Works:

http://www.ibm.com/developerworks/libra ry/wa-introhdfs/

• Jacob Leverich, C. K. (2010). On the Energy (In)eciency of Hadoop Clusters.

SIGOPS Operating Systems Review , 44

(1), 61-65.

• Jeffrey Dean, S. G. (2008). MapReduce: Simplified Data Processing on Large Clusters. (pp. 107–113). New Yark: Communications of the ACM.

• K. Chen, J. P. (2014). CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds. IEEE Transactions on Parallel and

Distributed Systems , 1403 – 1412, 2014.

• K. Morton, A. F. (2010). Estimating the progress of MapReduce pipelines. Data Engineering (ICDE), 2010 IEEE 26th International Conference (pp. 681–684). IQui

• Kamal Kc, K. A. (2010). Scheduling hadoop jobs to meet deadlines . CLOUDCOM '10 Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science (pp. 388-392). Washington DC: IEEE Computer Society .

• White, T. (2015). Hadoop:The Definitive Guide . (p. 6). O'Reilly Media.

• X. Cui, X. Lin. (2013). Modeling the Performance of MapReduce under

Resource Contentions and Task Failures.

Cloud Computing Technology and Science (CloudCom), (pp. 158–163). IEEE 5th International Conference.

• X. Lin, Z. M. (2012). A Practical Performance Model for Hadoop

MapReduce. Cluster Computing