International Journal of Scientific & Engineering Research, Volume 4, Issue 4, April -2013
ISSN 2229-5518

Mapreduce Performance in Heterogeneous Environments: A Review

Salma Khalil, Sameh A.Salem, Salwa Nassar and Elsayed M.Saad

Abstract— Mapreduce has become an important distributed processing model for large-scale data-intensive application like data mining and web indexing. Hadoop, an open-source implementation of Mapreduce, is widely used for short jobs requiring low response time. Mapreduce and Hadoop do not fundamentally consider heterogeneity of node and workload running in computer clusters. The current Hadoop implementation assumes that computing nodes in the cluster are homogeneous in nature. In this article, we survey some of the approaches that have been designed to improve the Mapreduce performance in heterogeneous environments.

Index Terms — Mapreduce, Cloud computing, Heterogeneous Environments, Hadoop, Distributed Computing, Data Locality, Fault Tolerance.

1 Introduction

An increasing number of popular applications become data-intensive in nature. In the past decade, the World Wide Web has been adopted as an ideal platform for developing data-intensive applications, since the communication paradigm of the Web is sufficiently open and powerful. Data-intensive applications like data mining and web indexing need to access ever-expanding data sets ranging from a few gigabytes to several terabytes or even petabytes. The leading example is Google, which uses its Mapreduce framework to process 20 petabytes of data per day. Mapreduce is an attractive model for parallel data processing in high-performance cluster computing environments. Mapreduce runs on a large cluster of commodity machines. Mapreduce offers fault tolerance that is entirely transparent to the programmers [1].

Hadoop [2] - a popular open-source implementation of the Mapreduce model is primarily developed by Yahoo, where it runs jobs that produce hundreds of terabytes of data on at least 10,000 cores [3]. Hadoop is also used at Facebook and Amazon [4].

The Mapreduce model runs on a large cluster consists of homogeneous nodes also assumes homogeneous workload when making a scheduling decision. Mapreduce takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, fault tolerance and managing the required inter-machine communication. The Mapreduce performance depends on the previous features which appear obviously in the homogeneous environment. The homogeneity assumption has been broken as:

- It is not always possible or even desirable to have a large cluster consists of single type of machine.
- It is not satisfied in virtualized data centers.
- It does not take the difference of workload characteristics between jobs into account when making a scheduling decision.

Thus, the need of employing the Mapreduce model on a heterogeneous environment becomes necessary. The heterogeneity affects the performance of the Mapreduce algorithm. Many researches [6], [7], [8], [9] and [13] has discussed how the heterogeneity affects the Mapreduce performance and developed algorithms to improve the performance of Mapreduce in the heterogeneous environments. In this article, we survey some of the algorithms that designed for improving the Mapreduce performance in heterogeneous environments, and examining their performance along with their relative strengths and weaknesses.

The rest of this article is organized as follows. Section II introduces an overview of the Mapreduce programming model, a brief introduction to Hadoop, an overview of the Mapreduce approaches in heterogeneous environment. Section III summarizes the approaches that have been developed to improve the data locality management of the Mapreduce model in heterogeneous environment. Section IV summarizes the approaches that have been developed to improve the fault tolerance support of the Mapreduce model in heterogeneous environment. Section V introduces a discussion about these approaches. Section VI concludes the article.

- *Salma Khalil is a Research Assistant at Electronic Research Institute and is currently preparing a master degree in Computer Engneering at Helwan University, Egypt. E-mail: salma_saber@eri.sci.eg*
- *Sameh A.Salem is Assistant Professor in Computer Engineering at Helwan University, Egypt.*
- *Salwa Nassar is the Head of HPCloud team at Electronic Research Institute, Egypt.*
- *Elsayed M.Saad is a Professor in Computer Engineering at Helwan University, Egypt.*

2 Background

2.1 Mapreduce Overview

The Mapreduce model was developed by Google [1] to run data-intensive applications on a distributed infrastructure like commodity cluster. Mapreduce was inspired by the map and reduce primitives present in Lisp and many other functional languages [1]. Mapreduce enables programmers with no spe-

International Journal of Scientific & Engineering Research Volume 4, Issue 4, April -2013
ISSN 2229-5518

cific knowledge of distributed programming to create her/his Mapreduce functions running in parallel across multiple nodes in the cluster by specifying two fundamental functions: a map function that processes key/value pairs to generate a set of intermediate key/value pairs, and a reduce function merges all the intermediate values associated with the same intermediate key. Mapreduce executes the map and reduce functions in parallel across the nodes of the cluster.

Programmers only need to implement the map and reduce functions, because a Mapreduce programming framework can facilitate some operations (e.g., grouping and sorting) on a set of key/value pairs. Mapreduce model is simple, because the programmers just have to focus on data processing functionality rather than parallelism details.

2.2 Hadoop

Hadoop's implementation of MapReduce closely resembles Google's [2]. There is a single master managing a number of slaves. The input file, which resides on a distributed file system throughout the cluster, is split into even-sized chunks replicated for fault-tolerance. Hadoop divides each MapReduce job into a set of tasks. Each chunk of input is first processed by a map task, which outputs a list of key-value pairs generated by a user defined map function. Map outputs are split into buckets based on key. When all maps have finished, the reduce tasks apply a reduce function to the list of map outputs with each key. Fig. 1 illustrates a MapReduce computation.

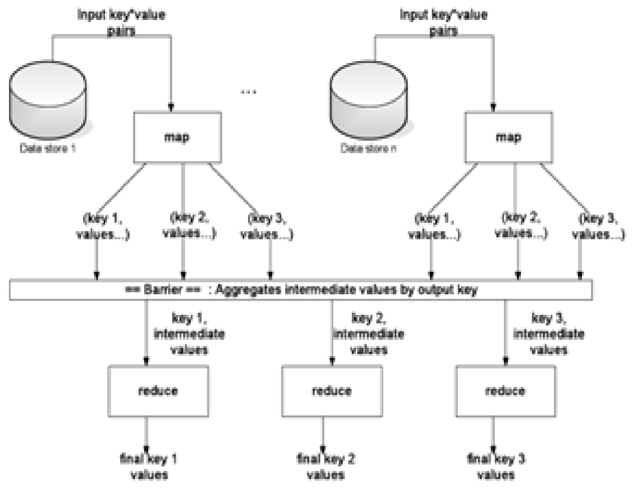


Fig. 1. A MapReduce computation [6].

The Hadoop framework consists of two main components: the MapReduce language and the Hadoop's Distributed File System (HDFS). Hadoop runs over HDFS, in which a file is divided into blocks and replicas of each block are stored on nodes in the cluster. Typically the block size is 64MB and each block has three replicas [5].

Hadoop assumes homogeneous clusters; which means that all nodes in the clusters have the same processing power and capability. Therefore, all nodes can finish the computation roughly at the same time. If a machine is slower than the other machines, it is treated as a faulty machine. In order to run the data-intensive applications, it is required to build very large clusters. Unfortunately, it is not always possible to have a large cluster with single type of machines. Consequently, this assumption is broken when deploying Hadoop in a heterogeneous cluster.

Hadoop assumes homogeneous workloads. In other words, it does not take the difference of the workload characteristics between jobs into account when making a scheduling decision. Also Hadoop is not satisfied in virtualized data centers.

2.3 Mapreduce Approaches in Heterogeneous Environment

Some researchers have discussed the degradation of Mapreduce performance in heterogeneous environments and presented solutions to improve its performance. Each approach improves one of the Mapreduce features that become a defect when deploying Mapreduce in a heterogeneous cluster. Therefore, the algorithms that represent each approach with respect to the improved feature are summarized into two categories as shown below:

I. Data Locality Algorithms.

II. Fault Tolerance Algorithms.

3 Data locality Algorithms

Data locality is a determining factor for the Mapreduce performance. In this section, the algorithms that have been developed to improve data locality management in a heterogeneous Hadoop cluster will be discussed.

3.1 Data Placement in Heterogeneous Hadoop Clusters

Hadoop distributes data to multiple nodes based on disk space availability [2]. This data placement strategy is efficient for a homogeneous environment where all the nodes have the same computing and disk capacity. In heterogeneous Hadoop cluster, a high-performance node can complete processing local data faster than low-performance node. After the fast node finished processing data residing in its local disk, the fast node has to handle the unprocessed data in remote slow node. The overhead of transferring unprocessed data from slow node to fast node is high if the amount of transferred data is huge. An approach to improve MapReduce performance in heterogeneous environments is to reduce the amount of data moved between slow and fast nodes in a heterogeneous cluster.

J.Xie et al. [7] developed a data placement mechanism in HDFS that distributed and stored a large data set across multiple heterogeneous nodes in accordance to the computing capacity of each node. In other words, the number of file fragments which distributed by data placement scheme and placed on the disk of each node is proportional to the data processing speed of each node.

This data placement algorithm implemented and incorporated two algorithms into Hadoop's HDFS. First, the initial data placement algorithm which initially distributed the file fragments to the heterogeneous nodes according to their computing capacities. Second, the data redistribution algorithm which reorganized the file fragments to solve the data skew problem.

Before implementing the data placement algorithm, they needed to measure the heterogeneity of a Hadoop cluster in terms of data processing speed. Such heterogeneity measurements in the cluster may change while executing different Mapreduce applications because the processing speed is highly depends on data-intensive applications. They introduced a metric - called computing ratio - to measure each node processing speed in a heterogeneous cluster.

The initial data placement algorithm begins first by dividing a large input file into a number of even-sized fragments. The responsibility of distributing the file fragments across the nodes of the cluster is governed by a data distribution server. It applies the round-robin algorithm to assign the input file fragments to the heterogeneous nodes based on their computing ratios. A small value of computing ratio indicates a high speed of node, meaning that the fast node must process a large number of fragments. Also a large value of computing ratio of a node indicates a low speed of the node, meaning that the slow node must process a small number of file fragments.

Input file fragments distributed by the initial data placement algorithm might be disrupted due to the following reasons: (1) new data is appended to an existing input file; (2) data blocks are deleted from the existing input file; and (3) new data computing nodes are added into an existing cluster. They implemented the data redistribution algorithm to reorganize the file fragment based on computing ratio.

The performance of their data placement mechanism in a heterogeneous Hadoop cluster is evaluated by using two data-intensive applications – Grep and WordCount. Table 1 summarized the parameters of the heterogeneous nodes used in tested cluster.

Table 1: Five Nodes in a Hadoop Heterogeneous Cluster [7].

Node	CPU Model	CPU(hz)	L1 Cache(KB)
Node A	Intel Core 2 Duo	2×1G=2G	204
Node B	Intel Celeron	2.8G	256
Node C	Intel Pentium 3	1.2G	256
Node D	Intel Pentium 3	1.2G	256
Node E	Intel Pentium 3	1.2G	256

The computing ratios for the heterogeneous Hadoop cluster are calculated with respect to Grep and WordCount (shown in Table2).

Table 2: Computing Ratios for the Grep and WordCount Applications [7].

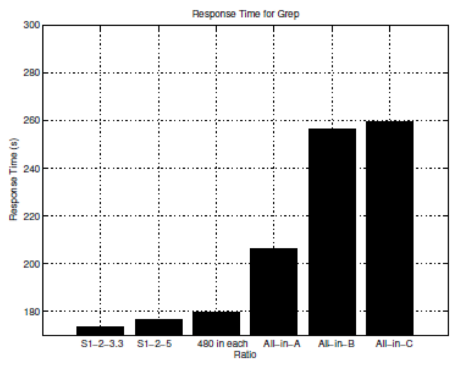
Computer Node	Ratios for Grep	Ratios for WordCount
Node A	1	1
Node B	2	2
Node C	3.3	5
Node D	3.3	5
Node E	3.3	5

Six data placement decisions are used (as shown in Table3) to evaluate their work.

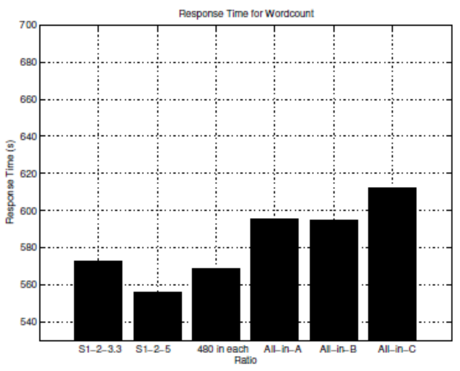
Fig. 2 shows the result of applying the decisions shown in Table 3 with data placement algorithm. Fig. 2(a) shows that S1-2-3.3 in Table 3 is the optimal decision for Grep to distribute data to the nodes of the tested cluster. Also, Fig. 2(b) shows that S1-2-5 in Table 3 is the optimal decision for WordCount.

Table 3: Six Data Placement Decisions [7].

Notation	Data Placement Decisions
S1-2-3.3	Distributing files under the computing ratios of the grep. (This is an optimal data placement for Grep)
S1-2-5	Distributing files under the computing ratios of the wordcount. (This is an optimal data placement for WordCount)
480 in each	Average distribution of files to each node.
All-in-A	Allocating all the files to node A.
All-in-B	Allocating all the files to node B.
All-in-C	Allocating all the files to node C.



(a)



(b)

Fig. 2. Performance of data placement algorithm with respect to: (a) Grep and (b) WordCount [7].

3.2 Data Locality Aware Task Scheduling Method for Heterogeneous Environments

In this research, the work is built upon the method to improve data locality of Mapreduce in homogeneous computing environments [10]. The method assumed that all nodes processing tasks have similar speed when selecting the node to issue the request. If the input data of a task is stored on the node, the method reserves the task for the node. However, this assumption cannot be held in the cloud computing, because there are many factors that can change the processing speed of the processors such as, the heterogeneity of the computational resources and its dynamic workload. Consequently, there is a need to develop an effective heterogeneity-oriented scheduling algorithm to improve the data locality management for Mapreduce model.

X.Zhang et al. [9] introduced a data locality aware scheduling method for heterogeneous Hadoop cluster. There are two factors affect the efficiency of map tasks execution; waiting time – is the shortest time that the task has to wait before it can be scheduled to one of the nodes that have the input data, transmission time – is the time needed to copy the input data of the task to the requesting node.

The objective is to make a tradeoff between the waiting time and transmission time at runtime when schedule a task to a node to obtain the optimal task execution time. After receiving a request from a requesting node, the method first schedules the task whose input data is stored on the requesting node. If there is no such kind of tasks, the method first selects the task whose input data is stored in the nearest node with respect to the requesting node. Then the method computes the waiting time and the transmission time of the selected task. If the waiting time is shorter than the transmission time, the method reserves the task for the node having the input data. Otherwise, it schedules the task to the requesting node.

A node can execute more than one task simultaneously. A node can issue a request whenever it completes a task and successive tasks have to wait for scheduling until the node completes its current tasks. Thus, the waiting time of the tasks whose input data are stored on the node can be represented by the shortest remaining time of all the tasks executing in the same node. Tasks are scheduled to the requesting node according to their probabilities.

The remaining time of the task can be calculated by the following equation [9]:

$$f(X,Y) = \begin{cases} \dfrac{n(1-x_{n-1})\prod_{i=0}^{n-1} y_i}{\sum_{i=0}^{n-1} x_i \prod_{j=0\ and\ j\neq i}^{n-1} y_j} & \forall x_i \in X, 0 < x_i < 1 \\ 0 & \exists x_i \in X, x_i = 0 \end{cases}$$

The scheduling method has been implemented in Hadoop-0.20.2. The topology of the tested cluster is illustrated by Fig. 3. Three evaluation criteria have been used to evaluate the system performance: (i) The number of the map tasks not scheduled to the nodes with the input data; (ii) The normalized execution time; and (iii) The response time of jobs. It is desirable that the system can obtain values smaller than the values of the default method.

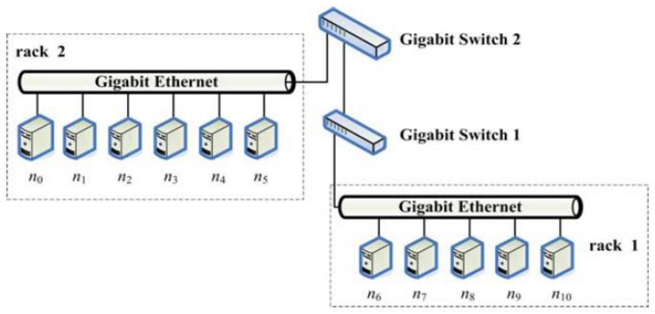


Fig. 3. The topology of the tested cluster [9].

Table 4 shows the details of the jobs. Two scenarios are introduced to execute these jobs. In scenario 1, the maximum of the tasks concurrently running on a node is configured with 2. In scenario 2, the maximum is configured with 3. Only the result of scenario 1 will be shown in this article.

Table 4: The details of the jobs [9].

Job Name	Block Size of HDFS(MB)	Input Data Size GB	Map Tasks	Reduce Tasks
I	64	2.6	41	2
II	128	5.0	41	2
III	256	10.1	41	2

Comparing with the default Hadoop, the mean number of jobs I and II are reduced by 13% and 4 % respectively in (shown in Fig. 4). In the best case, the normalized execution time is reduced by 12%. Also, Hadoop with the proposed method achieved shorter response time.

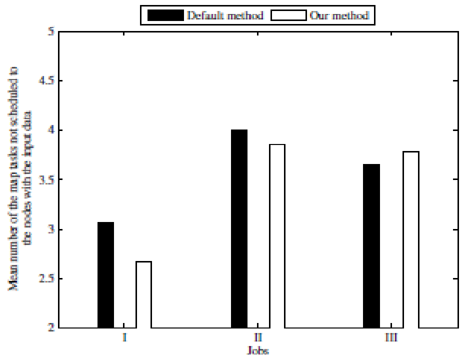


Fig. 4. The number of the tasks not scheduled to the nodes with the input data in scenario 1 [9].

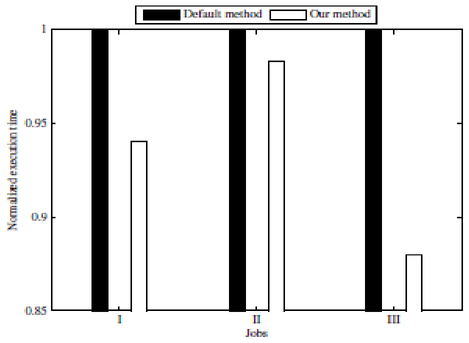


Fig. 5. The normalized execution times of jobs I, II and III in scenario 1 [9].

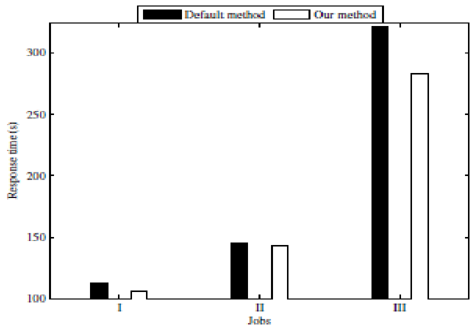


Fig. 6. The response times of jobs I, II and III in scenario 1 [9].

International Journal of Scientific & Engineering Research Volume 4, Issue 4, April -2013
ISSN 2229-5518

4 FAULT TOLERANCE ALGORITHMS

A key benefit of Mapreduce is that it automatically handles failures and hides the complexity of the fault tolerance from the programmers.

Hadoop's performance is closely tied to its task scheduler, which assumes that the cluster nodes are homogeneous and tasks make progress linearly. Hadoop's scheduler uses these assumptions to decide when to speculatively re-execute tasks that appear to be stragglers. Hadoop's scheduler starts speculative tasks based on a simple heuristic comparing each task's progress to the average progress. This heuristic works well in the homogeneous environments where the stragglers are obvious. Hadoop's scheduler can cause server performance degradation in heterogeneous environments because the underlying assumptions are broken.

In this section, the algorithms that have been developed to improve fault tolerance support in the heterogeneous Hadoop cluster will be discussed.

4.1 LATE: Longest Approximate Time to End Algorithm

M.Zaharia et al. [6] designed a new scheduling algorithm for speculative execution, LATE algorithm, which is highly robust to heterogeneity. LATE reduces Hadoop's response time by a factor of 2. LATE is based on three principles: prioritizing tasks to speculate, selecting fast nodes to run on, and capping speculative tasks to prevent thrashing.

When a node has an empty task slot, Hadoop chooses a task for it from one of three categories. First, any failed tasks are given the highest priority. Second, non-running tasks are considered, specially the map tasks that have local data on this node. Third, the tasks which need to execute speculatively.

Hadoop monitors task progress using progress score between 0 and 1 to select speculative tasks. For a map task, the progress score is the fraction of the input data read. For a reduce task, the execution is divided into three phases (copy phase, sort phase, reduce phase), each of which represents 1/3 of the progress score. Hadoop defines a threshold for speculative execution using the average progress score of each category of tasks (maps and reduces). When a task's progress score is less than the average off its category minus 0.2, and the task has run at least one minute, it is marked as a straggler.

LATE always speculatively executes the task which will finish farthest in the future. LATE estimates the task's finish time based on the progress score provided by Hadoop. Hadoop estimates the progress rate of each task as ProgressScore/T , where T is the amount of time the task has been running for, and then estimate the task's finish time as (1−ProgressScore)/ProgressRate.

LATE performance is evaluated using two environments: large cluster on Amazon Elastic Computing Cloud (EC2) [11] and a local virtualized testbed. Fig. 7 shows the response time achieved by each scheduler in the heterogeneous EC2 cluster. Fig. 8 shows the response time by each scheduler when stragglers exist.

4.2 SAMR: A Self-Adaptive Mapreduce Scheduling Algorithm

LATE scheduling algorithm [6] takes the heterogeneity assumptions into consideration, but has poor performance due to the static manner in computing the progress of the tasks. Consequently, neither Hadoop nor LATE schedulers are desirable in heterogeneous environment.

Q.Chen et al. [8] proposed SAMR scheduling algorithm, which calculates the progress of the tasks dynamically. SAMR adopted the idea of LATE scheduling algorithm and proposed a modified version to improve Mapreduce in terms of saving the time of the execution and the system's resources. SAMR decreases the time of the execution up to 25% compared with Hadoop's scheduler and 14% compared with LATE scheduler.

SAMR is using historical information recorded on each node to tune parameters and find slow tasks dynamically. SAMR updates the values after every execution. SAMR is taking the two stages characteristic of map tasks into consideration. SAMR is classifying slow nodes into map slow nodes and reduce slow nodes.

SAMR performance is evaluated using virtual machines built upon a cluster of five personal computers.

Fig. 9 shows the efficiency of SAMR when running Sort benchmark. The execute time of Hadoop is used as the baseline. Hadoop without backup mechanism spends about double time in executing the same job. LATE decreases about 7% execute time, LATE using historical information mechanism decreases about 15% execute time, SAMR decreases about 24% execute time compared to Hadoop.

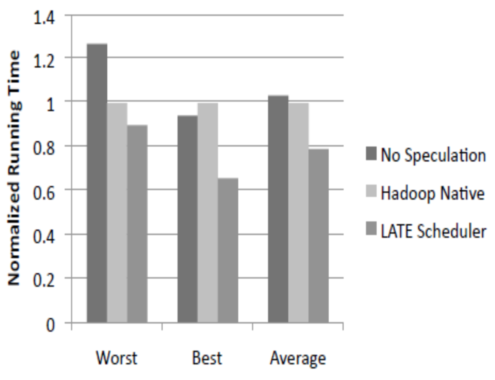


Fig. 7. EC2 sort running in heterogeneous cluster [6].

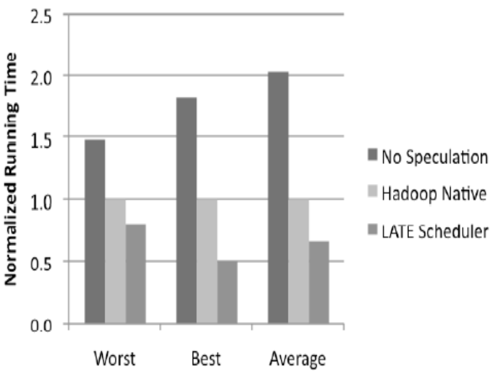


Fig. 8. EC2 sort running with stragglers [6].

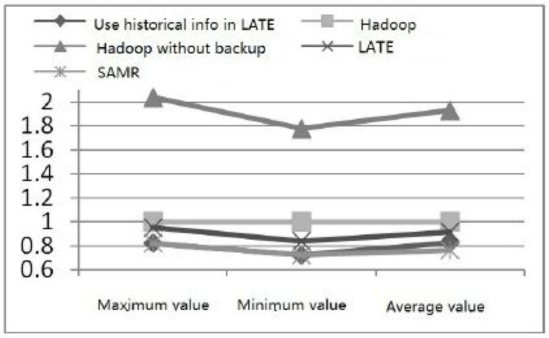


Fig. 9. The results of running the sort benchmark on the experiment platform [8].

5 Discussion

This article reviews some of the approaches that have been developed to improve the Mapreduce performance in the heterogeneous environments. The features of the Mapreduce model perform poorly in a heterogeneous computing cluster.

Data locality management is one of the features that have worse affect on the Mapreduce performance [12]. Bad data locality management increases the execution time of the Mapreduce job. Some of approaches have been recently introduced [7], [9] to improve the data locality management in heterogeneous Mapreduce clusters. These approaches have some limitations. The first approach [7] of data locality management did not consider the data replication. If a node crashes, its input files are lost because there was no duplication of the input files. Additionally, it does not automatically support the fault tolerance feature. This approach can be enhanced by handling the redundancy issue of data allocation in the cluster, and designing a dynamic distribution mechanism for multiple data intensive applications. The second approach [9] of data locality management did not consider the workload heterogeneity. This approach can be enhanced by developing more effective technique for transmitting the data sets in the Hadoop environments. Also, this approach can be enhanced by a more accurate estimation of the transmission time.

Another feature that has bad effect on the Mapreduce performance is the fault tolerance. If this feature is not considered, this may need re-submitting the whole job that has one or more unaccomplished tasks. Some of recent approaches have been proposed [6], [8] to improve the fault tolerance in heterogeneous Mapreduce clusters. LATE algorithm [6] and SAMR [8] algorithm do not consider the data locality management for launching backup tasks. The LATE algorithm is estimating the time left to detect the faulty tasks. This method does not work properly when the tasks slow down. Different methods can be used to accurately estimate the time left in order to enhance the LATE algorithm. Also, estimating completion time for each phase in the reduce task independently can enhance the LATE algorithm. For SAMR algorithm, a better mechanism for tuning the parameters can be introduced to enhance Mapreduce performance.

It should be noted that the integration of data locality with fault tolerance approaches can improve the overall Mapreduce performance in the heterogeneous environments. This integration will gather the advantages of these approaches and enhance the Mapreduce performance in the heterogeneous clusters. Therefore, there is a need for an approach for enhancing more than one feature of the Mapreduce model in the heterogeneous clusters. This will lead to have a Mapreduce model in the heterogeneous environments with performance similar to the performance of the Mapreduce model in the homogeneous environments.

6 Conclusion

In this article, an investigation of the key features that affect Mapreduce performance in heterogeneous environments is presented. These include data locality and fault tolerance. As demonstrated, some of recent approaches are presented along with a further discussion on their relative strengths and weaknesses. Also, some enhancements that can be developed to improve the performance of the Mapreduce model in the heterogeneous environments have been highlighted. As discussed, there is a need to integrate data locality and fault tolerance approaches to improve the overall Mapreduce performance in heterogeneous computing clusters. This integration may provide an acceptable performance for the Mapreduce model in heterogeneous environments.

References

[1] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[2] Hadoop, http://lucene.apache.org/hadoop, (last view June 30, 2012).

[3] Yahoo! launches worlds largest hadoop production application, http://developer.yahoo.com/blogs/hadoop/posts/2008/02/yahoo-worlds-largest-production-hadoop/, (last view June 30, 2012).

[4] Applications powered by Hadoop, http://wiki.apache.org/hadoop/PoweredBy, (last view June 30, 2012).

[5] K.Shvachko, H.Kuang, S.Radia, R.Chansler. The Hadoop Distributed File System. In proceeding the 26th IEEE Symposium on Massive Storage Systems and Technologies, 2010.

[6] M.Zaharia, A.Konwinski, A.Joseph, Y.zatz, and I.Stoica. Improving mapreduce performance in heterogeneous environments. In OSDI'08: 8th USENIX Symposium on Operating Systems Design and Implementation, October 2008.

[7] J.Xie, S.Yin, X.Ruan, Z.Ding, Y.Tian, J.Majors, A.Manzanares, and X.Qin. Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters. In proceedings of IEEE International parallel and distributed Processing Symposium, 2010.

[8] Q.Chen, D.Zhang, M.Guo, Q.Deng and S.Guo. SAMR: A Self-adaptive MapReduce Scheduling Algorithm In Heterogeneous Environment. In Proceedings of IEEE 10th International Conference on Computer and Information Technology, 2010.

[9] X.Zhang, Y.Feng, S.Feng, J.Fan and Z.Ming. An Effective Data Locality Aware Task Scheduling Method for MapReduce Framework in Heterogeneous Environments. In International Conference on Cloud and Service Computing, 2011.

[10] X. Zhang, Z. Zhong, B. Tu, S. Feng, and J. Fan. Improving data locality of mapreduce by scheduling in homogeneous computing environments. In Proceedings of IEEE 9th International Symposium on Parallel and Distributed Processing with Applications, pages 120–126, Busan, Korea, 2011. IEEE.

[11] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2, (last view June 30, 2012).

[12] Z.Guo, G.Fox, M.Zhou. Investigation of Data Locality in MapReduce. In Proceeding of IEEE/ACM 12th International Symposium on Cluster, Cloud and Grid Computing, 2012.

[13] Z.Guo, G.Fox. Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization. In proceeding of IEEE/ACM 12th International Symposium on Cluster, Cloud and Grid Computing, 2012.

[14] Fourth Ann. Allerton Conf. Circuits and Systems Theory, pp. 8-16, 1994. (Conference proceedings)

[15] H. Goto, Y. Hasegawa, and M. Tanaka, "Efficient Scheduling Focusing on the Duality of MPL Representation," Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS '07), pp. 57-64, Apr. 2007, doi:10.1109/SCIS.2007.367670. (Conference proceedings)

[16] J. Williams, "Narrow-Band Analyzer," PhD dissertation, Dept. of Electrical Eng., Harvard Univ., Cambridge, Mass., 1993. (Thesis or dissertation)

[17] E.E. Reber, R.L. Michell, and C.J. Carter, "Oxygen Absorption in the Earth's Atmosphere," Technical Report TR-0200 (420-46)-3, Aerospace Corp., Los Angeles, Calif., Nov. 1988. (Technical report with report number)

[18] L. Hubert and P. Arabie, "Comparing Partitions," J. Classification, vol. 2, no. 4, pp. 193-218, Apr. 1985. (Journal or magazine citation)

[19] R.J. Vidmar, "On the Use of Atmospheric Plasmas as Electromagnetic Reflectors," IEEE Trans. Plasma Science, vol. 21, no. 3, pp. 876-880, available at http://www.halcyon.com/pub/journals/21ps03-vidmar, Aug. 1992. (URL for Transaction, journal, or magzine)

[20] J.M.P. Martinez, R.B. Llavori, M.J.A. Cabo, and T.B. Pedersen, "Integrating Data Warehouses with Web Data: A Survey," IEEE Trans. Knowledge and Data Eng., preprint, 21 Dec. 2007, doi:10.1109/TKDE.2007.190746.(PrePrint)